Performance Evaluation of Software Defined Networks Using Queueing Models

Maysarah Mohammad Al Masri

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master in Science
in
Computer Engineering

Eastern Mediterranean University
May 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Doğu Arifler
Supervisor

Examining Committee

1. Prof. Dr. Doğu Arifler

2. Assoc. Prof. Dr. Enver Ever

3. Assoc. Prof. Dr. Mohammed Salamah

ABSTRACT

Evolving network requirements have recently made the software defined networking paradigm very popular. In a software defined network (SDN), the data and control functions of network nodes such as routers and switches are separated. In particular, a physically separate controller, which is implemented in software, computes and distributes forwarding tables to routing devices. Such a separation requires an analysis of packet delay performance to evaluate the tradeoffs of using controllers versus a traditional networking architecture. Previous studies have employed simulations and analytical models to evaluate the performance of SDNs before actual deployment. However, these studies are limited to specific topologies, are based on approximations, and cannot be easily extended to more general topologies.

The work presented in this thesis employs classed networks of queues to model SDNs. First, a topology that consists of a single switch and a single controller is analyzed using the proposed queueing model. Then, the topology is extended to multiple switches and the methodology is applied to model the extended network. Finally, the classed queueing network model is used to evaluate the deployment of multiple controllers. The single-switch, single-controller topology results are in agreement with previous studies that employ single-class queueing theoretic methods. There is currently limited or no data available to benchmark classed queueing network models of multiple-switch or multiple-controller topologies. Nevertheless, the results give insights into the design and deployment of multiple switches or controllers. For instance, the findings indicate that the power-delay performance is improved when two half-capacity controllers are deployed instead of a single

full-capacity controller. In addition, in case there are intermittent controller failures, installing two controllers may be justified for large traffic loads.

Keywords: Classed network of queues, OpenFlow, Performance analysis, Queueing theory, Software defined networks

ÖZ

Gelişen ağ gereksinimleri, yazılım tanımlı ağ (SDN) yaklaşımlarını oldukça popüler hale getirmiştir. SDN'lerde, yönlendirici ve anahtar gibi ağ düğümlerinde, veri ve kontrol fonksiyonları birbirinden ayrılmıştır. Özellikle, yazılım olarak uygulanan ve fiziksel ayrı bir kontrol birimi, gönderim tablolarını hesaplayıp yönlendirici cihazlara dağıtır. Bu ayırım, geleneksel ağ mimarisinden farklı olduğundan, kontrol birimi kullanmanın analiz edilmesini gerektirmektedir. Önceki çalışmalar, simülasyon ve analitik modellerle kurulum öncesi SDN'lerin performansını değerlendirmiştir. Ancak, bu çalışmalardaki metotlar bazı spesifik topolojilerle sınırlı olup, yaklaştırmalara dayalıdır ve genel topolojilere genişletilmesi kolay değildir.

Bu tezde anlatılan çalışma, sınıflı kuyruklar ağlarını kullanıp SDN'leri modellemeyi amaçlamaktadır. İlk olarak, bir anahtar ve bir kontrol biriminden oluşan topoloji önerilen kuyruk modeliyle analiz edilmiştir. Daha sonra, topoloji birden fazla anahtarla genişletilmiş ve metotlar bu topolojiye uygulanmıştır. Son olarak, sınıflı kuyruklar ağ modeli, birden fazla kontrol biriminin kurulumunu değerlendirmek için kullanılmıştır. Tek-anahtar, tek-kontrol biriminden oluşan topolojiyle elde edilen sonuçlar daha önceki çalışmalarda elde edilen ve tek sınıflı kuyruk teorisi kullanan metotlarla uyuşmaktadır. Şu anda, bu konuda çalışmalar ve sonuçlar sınırlı olduğundan, birden fazla anahtar ve birden fazla kontrol birimi modellerinin verdiği sonuçlar teyit edilememiştir. Ancak, buna rağmen sonuçların birden fazla anahtar ve kontrol birimi içeren ağların tasarım ve kurulumuna yol göstereceği öngörülmektedir. Örneğin, sonuçlara göre, güç-gecikme performansının, tek bir tam kapasite kontrol birimi yerine iki tane yarım kapasiteli kontrol birimi kullanıldığında daha iyi olacağı

beklenmektedir. Ayrıca, ara sıra devre dışı kalan kontrol birimleri olması durumunda, iki kontrol birimi kurmanın, yüksek trafik yoğunluğunda tercih edilebileceği yönünde bulgular elde edilmiştir.

Anahtar Kelimeler: Sınıflı kuyruklar ağı, OpenFlow, Performans analizi, Kuyruk teorisi, Vazılım tanımlı ağlar

DEDICATION

Dedicated to my family.

ACKNOWLEDGMENT

My deepest gratitude goes to my supervisor Prof. Dr. Doğu Arifler for his constant support and wholehearted collaboration in my thesis. The list of skills I have learned from him is too long to be included here. Therefore, simply thanks for everything, for being a great teacher, for great research support, advice and for correcting this thesis. Thanks for guiding me through this time, it has been and will be an honor and great pleasure working with you.

My personal appreciation goes to Assoc. Prof. Dr. Enver Ever, and Assoc. Prof. Dr. Mohammed Salamah for accepting to be in my jury. Their valuable comments and suggestions have truly helped me to further improve my thesis writing and contents.

I am most grateful to my parents who were always there to support me and to share my thoughts. I want to thank my mother for her love and endless support, thanks for always being there for me and for being here in Cyprus during my defense. Thanks to all my brothers and sister who always encourage me to finish this long journey.

Yet, in the end, my special thanks go to my life partner and dearest person Hamza who reminded me that there is a life beyond the school. Without him I would not have made it here. Thanks for understanding that time was often scarce and for always encouraging me. Thanks for listening to me when I needed someone to talk to, thanks for all the nights we spent managing time and scheduling for the next-step goals.

After all, I believe that the research I did during my Master is just an exercise and the best is coming with time. This is just the beginning.

TABLE OF CONTENTS

ABSTRACT iii

ÖZ v

DEDICATION vii

ACKNOWLEDGMENT viii

LIST OF FIGURES x

LIST OF SYMBOLS AND ABBREVIATIONS xiii

1 INTRODUCTION 1

1.1 Software Defined Networking 1

1.1.1 SDN Architecture 2

1.2 OpenFlow (OF) 5

1.2.1 OpenFlow Overview 5

1.2.2 Packet Matching 7

1.3 Thesis Motivation 7

1.4 Related Work 9

1.5 Main Contributions 12

1.6 Thesis Structure 12

2 MODELS AND METHODS 13

2.1 Jackson Network Model of SDN 13

2.2 Modified Jackson Network 15

2.3 Classed Networks of Queues 17

2.4 Proposed Model for SDN Using Classed Networks of Queues 18

2.4.1 One Controller, One Switch Topology 19

2.4.2 Extended Data Plane Topology 20

2.4.3 Topology with Multiple Controller 21
2.4.4 Energy Performance Trade-off 23
3 RESULTS AND DISCUSSION 25
3.1 One-Controller, One-Switch Topology 26
3.2 Extended Data Plane 27
3.3 Multiple Controller Topology 28
3.3.1 Energy Performance Trade-off 30
3.3.2 Effect of Redundancy with Intermittently Failing Controllers 35
3.3.3 Benefits of Employing Multiple Controllers 36
4 CONCLUSIONS AND FUTURE WORK 38
4.1 Conclusions 38
4.2 Future Work 39
REFERENCES 45
APPENDIX 46

LIST OF FIGURES

Figure 1.1: SDN architecture [10] 2

Figure 1.2: Centralized controller [11] 3

Figure 1.3: Distributed controller [11] 4

Figure 1.4: General structure of SDN application plane functions [8] 4

Figure 1.5: Connectivity between OF switch and controller [16] 6

Figure 1.6: Flow table entry for OF version 1.0.0. 6

Figure 1.7: 12−tuples of matching fields. 8

Figure 1.8: Packet matching process 8

Figure 1.9: Queuing model of simple network with an OpenFlow switch connected to a single controller [23]. 10

Figure 2.1: An example Jackson queueing network model [29]. 13

Figure 2.2: Jackson queueing network model applied to SDN [19] 14

Figure 2.3: Modified Jackson model [19] 15

Figure 2.4: Simple classed network of queues 17

Figure 2.5: Flowchart of proposed model 18

Figure 2.6: Simple OF network of proposed model 19

Figure 2.7: Extended data plane 20

Figure 2.8: Extended control plane 22

Figure 3.1: Comparison of the proposed model to the modified Jackson model 26

Figure 3.2: Extended data plane with three switches 27

Figure 3.3: Comparison of different dispatching policies 29

Figure 3.4: Queue sizes for each of the two controllers under different dispatching policies 31

Figure 3.5: Cost function when $\alpha = 2$ 32

Figure 3.6: Cost function when $\alpha = 3$ 33

Figure 3.7: Controllers with intermittent failures 34

Figure 3.8: Failure all scenarios 36

LIST OF SYMBOLS AND ABBREVIATIONS

API	Application Programming Interface
FCFS	First Come First Served
ICN	Information Centric Networking
IoT	Internet of Things
JSQ	Join the Shortest Queue
NOS	Network Operating System
NBI	Northbound Interface
ONF	Open Networking Foundation
OF	OpenFlow
PDP	Power Delay Product
RR	Round-Robin
SSL	Secure Sockets Layer
SDN	Software Defined Networking
TLS	Transport Layer Security

Chapter 1

INTRODUCTION

1.1 Software Defined Networking

The rapid growth in network complexity as well as mobility and the wide use of Internet of Things (IoT) and virtualization have made network management difficult. Traditional networks require expert personnel to (re)configure and setup network devices and appliances; this process is time consuming, costly and can become unscalable in certain cases [1]. In addition, as the needs of users are increasing sharply, developers and companies implement and develop new applications and services in order to meet the customer requirements. Today's data centers, campuses, and enterprises need a more adaptable architecture, improve security and on demand scaling. These challanges have led network researchers to rethink about new designs of network architecture. The new designs should address the weaknesses in traditional networks and make them compatible with the new requirements [2].

Software Defined Networking (SDN) was initiated at Stanford University [3] and represents a new approach in computer networking. SDN provides network management simplification, energy efficiency, and adaptive scalability. Moreover, network resources can be installed, configured, and managed in a simple manner [4, 5]. The core idea of this new paradigm is the decoupling of the control plane (controller) and the data plane (switching devices). This decoupling allows

flexible management of the network resources on the fly instead of manual (re)configuration [6, 7]. The main contribution of the SDN is its ability to enable flexible and centralized control of the network.

1.1.1 SDN Architecture

Figure 1.1 illustrates the logical structure of a typical SDN. As defined by [8] the infrastructure layer, which includes the hardware devices like routers, switches, and access points, lies at the bottom layer. Infrastructure layer, forwarding elements, switching devices, network devices, and data plane all refer to the hardware devices in SDN. Infrastructure layer is responsible for forwarding the incoming packets via different ports based on flow table rules that are determined by the controller. It is also responsible for collecting network traffic and network usage statistics [9].

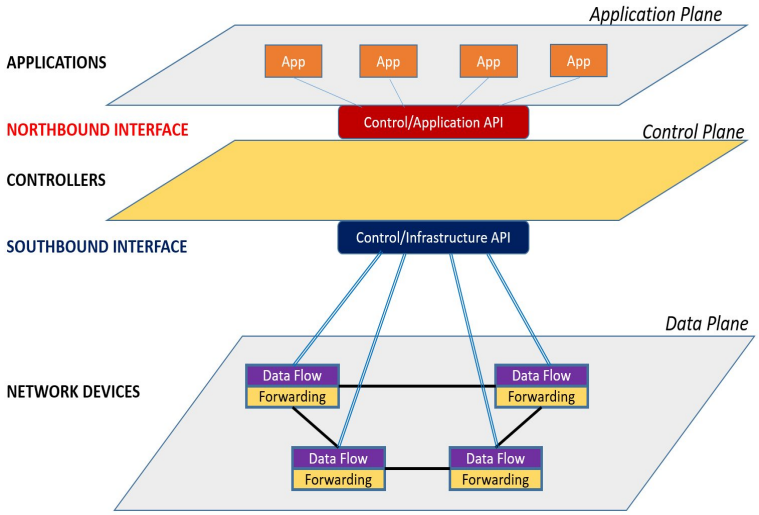


Figure 1.1: SDN architecture [10]

The control layer (central layer) is a functional-based layer which is responsible for switching, routing, mobility, and security. The network operating system (NOS) or controller is a critical and key element in the SDN and is considered as the brain of the network. The NOS has a global view of the network, determines the routes for

the flows, and sets the forwarding rules at networking devices on a path. Nowadays, there are different implementations of NOS available in the market using different programming languages such as C++, Java, and Python.

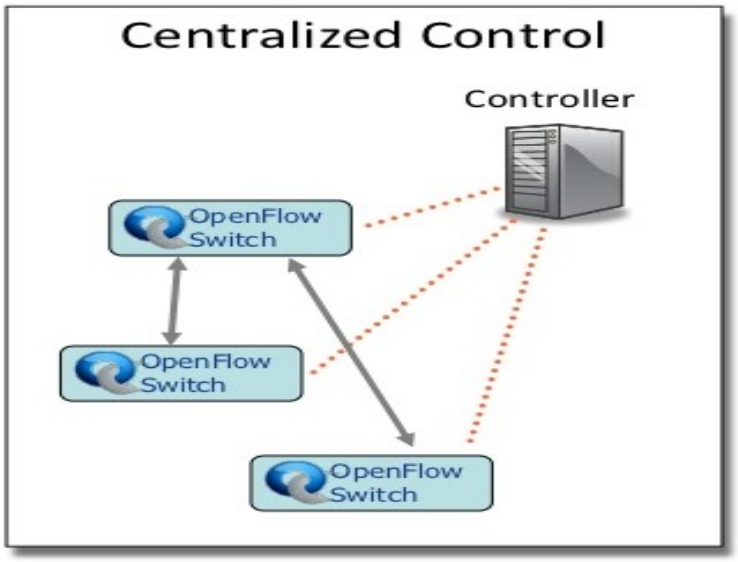


Figure 1.2: Centralized controller [11]

NOS can be classified as a centralized or a distributed architecture as shown in Figures 1.2 and 1.3, respectively. In centralized architecture, only one NOS masters the switching devices in infrastructure layer. This model represents a single point of failure, and has limited scalability. Whereas in a distributed architecture, there are many controllers which manage the forwarding devices. In this architecture, all the controllers should have the same topology information simultaneously with redundancy mechanisms for any kind of failure which may occur in the network.

The communication protocol between the control plane and the data plane is realized over the southbound application programming interface (API). One standard example of the southbound API is the OpenFlow (OF) protocol, which is a combination of specifications for setting the logical structure of a network. Moreover, OF can be

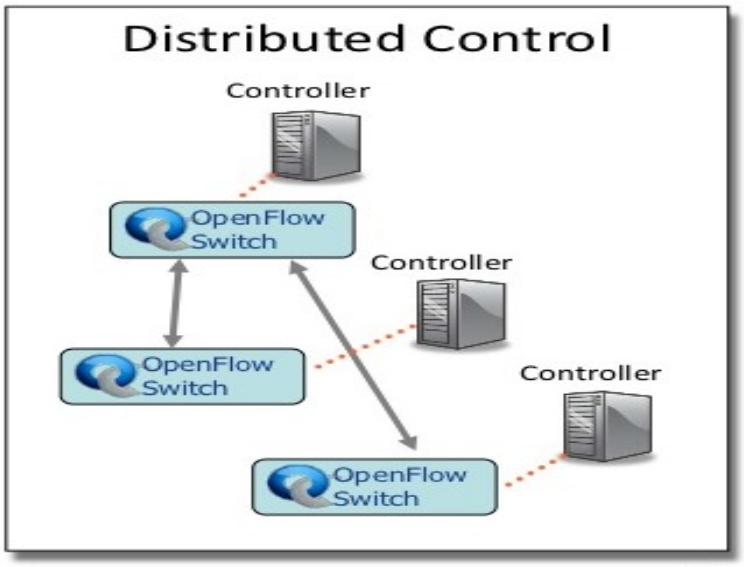


Figure 1.3: Distributed controller [11]

considered as a documented protocol between SDN control and data plane [8]. Further discussions about OF are given in Section 1.2.

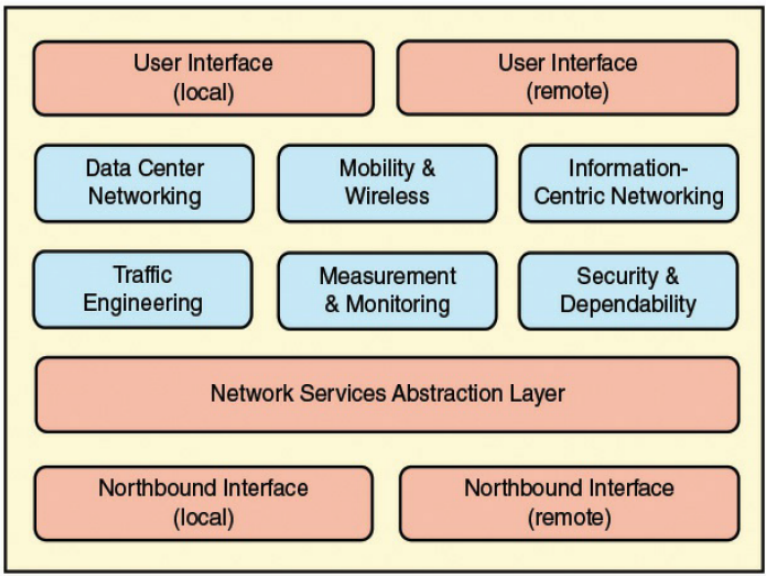


Figure 1.4: General structure of SDN application plane functions [8]

The application layer includes all the necessary services and operations required by the network. These applications determine, control, and monitor the network behavior and resources. Utilizing application control interfaces, the controller communicates

with the application layer in order to customize the behavior and the properties of the network resources. Figure 1.4 shows a set of examples for SDN application plane functions and interfaces. Interfaces can be local or remote, with one type used by the user which is called the "user interface", and the other coupled with the controller which is called the "northbound interface (NBI)". Variety of applications can be executed by the application layer such as data center, wireless networking, traffic engineering, and information centric networking (ICN).

1.2 OpenFlow (OF)

OpenFlow [12] is a protocol published by the Open Networking Foundation (ONF) and is used to apply the concept of SDN in real network applications [13]. It is used to interact between the controller(s) and the forwarding element(s). Initially, it was used in scholastic grounds, but then it started to be employed in commercial applications [12]. OpenFlow has different versions and it has special abilities: for instance, it is used to investigate traffic performance by software, and to control numerous routers from a master controller. Additionally, it has the ability to update the forwarding tables automatically. These capabilities enable innovative applications which are more secure and manageable. In this section, the discussion is based on the fundamental version of OF (version 1.0) [14]. Currently, there are six versions of OF, where additional features are added to the original protocol in order to improve performance and make it user-friendly. For examle, version 1.1 has multiple flow table and group table compared to version 1.0. Version 1.4 has a default port for the packets which can be sent if there is no match in the flow table [15].

1.2.1 OpenFlow Overview

The main components of OpenFlow are: 1) OpenFlow controller, 2) OpenFlow protocol, and 3) OpenFlow switch, as illustrated in Figure 1.5. In conventional

networks, the router runs a routing algorithm in order to decide the route to a particular destination. In OpenFlow, this is done by the controller which is responsible for programming all the packet forwarding rules in the switch.

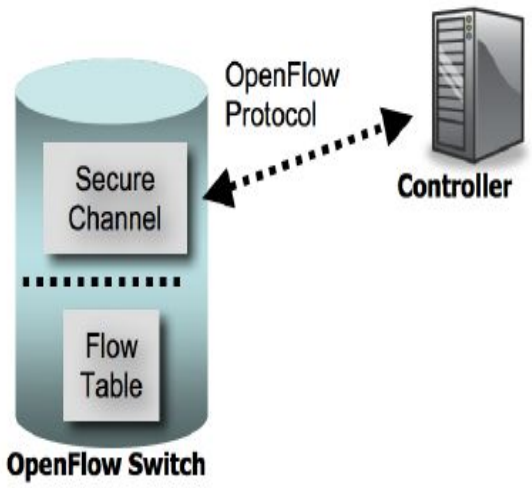


Figure 1.5: Connectivity between OF switch and controller [16]

The OF protocol defines the communication between the OpenFlow controller and the switch. The connection between the controller and the switch is secured by the secure sockets layer (SSL) or transport layer security (TLS) encryption protocols. The OF switch contains one or multiple flow tables and one group table. Each flow table contains flow entries, which consists of match fields, counters, and actions to be applied on the matched packets. The main function for the switch is to check the flow table entries to forward the packets.

Header Fields	Counters	Actions

Figure 1.6: Flow table entry for OF version 1.0.0.

Figure 1.6 shows the main components of the flow table entry which are listed as:

- **Header fields or match fields**: These are used to compare incoming packets header fields to decide if there is a match or not.
- **Counters**: These are used to keep track of statistics such as the number of packets/bytes forwarded or dropped.
- **Actions**: These identify how to deal with the packets in the flow.

1.2.2 Packet Matching

The flow entries are handled in order. Once a match is discovered, no further matches are made against that flow table. If all the flow tables are checked, but no match is found, this status is called a table miss. When the packet arrives to the OF switch, the header is examined against 12-tuple of match fields as shown in Figure 1.7. The fields can be set to wildcards, which means that all the packets will match the entry. If a packet match is found, the switch applies the actions and updates the statistics. Otherwise, the packet (or part of the packet) in the flow is encapsulated and sent to the controller to determine the status of the packet. The controller forwards the rule to all routers and switches on the path of the packet. The complete process is shown in Figure 1.8.

1.3 Thesis Motivation

SDNs are highly dynamic and scalable systems. They have to be evaluated under different workloads for a variety of topologies and redundancy mechanisms. As a result, neither real measurements nor packet level simulations will be fast enough and cost-effective for exploring all the different parameters and cases that need to be considered before deployment or reconfiguration of SDNs. This thesis aims to study and model the performance of SDN switches and controllers based on queuing theory and simulations in order to predict SDN-enabled network performance before actual

Matching fields
Switch input port
VLAN ID
VLAN priority
Ethernet source address
Ethernet destination address
Ethernet frame type
IP source address
IP destination address
IP protocol
IP Type of Service (ToS) bits
TCP/UDP source port
TCP/UDP destination port

Figure 1.7: 12−tuples of matching fields.

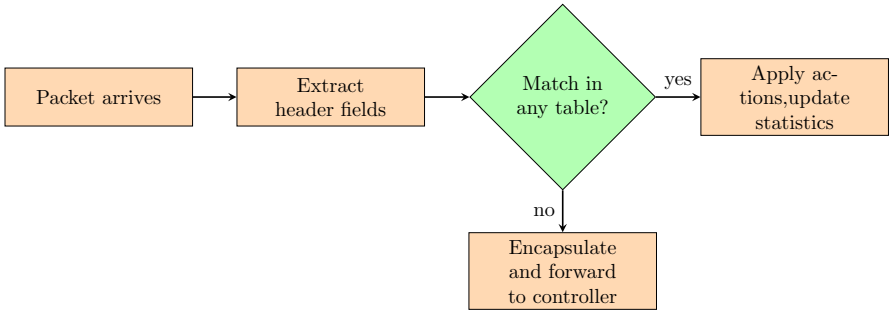


Figure 1.8: Packet matching process

deployment. Queueing models and simulations may rather rapidly provide designers with useful performance predictions for evolving SDN requirements. This study is reinforced by analytical solutions based on queuing theory in order to directly compare the simulation and the analaytical results. There are only a few studies evaluate the performance of SDN and OpenFlow networks. Among theses studies, some are described in Section 1.4; many studies are very specific to special topologies and for certain applications. Contrarily, the study provided in this thesis is more general and can be applied to multiple switch and controller topologies. The performance metric measured in the proposed model is the sojourn time of the packets, which measures how much time a packet spends in the network. The sojourn

time is measured in case of multiple controller and in case of multiple switch scenarios.

1.4 Related Work

In this section, a review is provided about some of the previous research studies that use queueing theory to evaluate controller-switch performance of SDN networks. Queueing methodology [17] considers the average performance quantities in steady state. In [18], an analytical study on OpenFlow network with Transmission Control Protocol (TCP) traffic is described. The configuration consists of a single controller and a switch, which is modeled as a feedback queuing system. The switch is modeled as an M/M/1 queue with infinite buffer size. The expected value of the packet service time is 9.8 μs and arrivals are modeled as a Poisson process. The proposed model is verified using Nox controller. The NOS queue is modeled as an M/M/1-S queue with capacity of 512 packets for exponentially distributed service times with different means. Simulation results show the correlation between the network sojourn time and controller service rate. Furthermore, the correlation of the switch service rate on the packet payload size and the flow rules within the flow table is demonstrated through simulations. The main drawbacks of the proposed model in [18] are: 1) the inability to extend the model to a multi-switch framework; and 2) traffic coming to a switch from the controller should not return to the controller again. The shortcoming in [18] is adderessed in [19]; more details about the work is discussed in Chapter 2.

The approach used in [20, 21] focuses on measuring the performance of NOS. The NOS is modeled as an M/G/1 queue and the switch as an $M^{[X]}/M/1$ queue. The authors derive the sojourn time for "PACKETIN" messages which are sent form the switch to the NOS when a table miss happens. Simulations using Floodlight with the benchmark tool Cbench demonstrate that forwarding rate of packets in large networks

depend on NOS processing capacity.

In [22], network elements are assumed to be M/Geo/1 queues with Poisson arrivals and service times geometrically distributed. Yet, the authors do not consider the controller-switch interaction. The results show the critical factors of mean response time as: 1) number of flow table entries and their positions; and 2) packet arrival times.

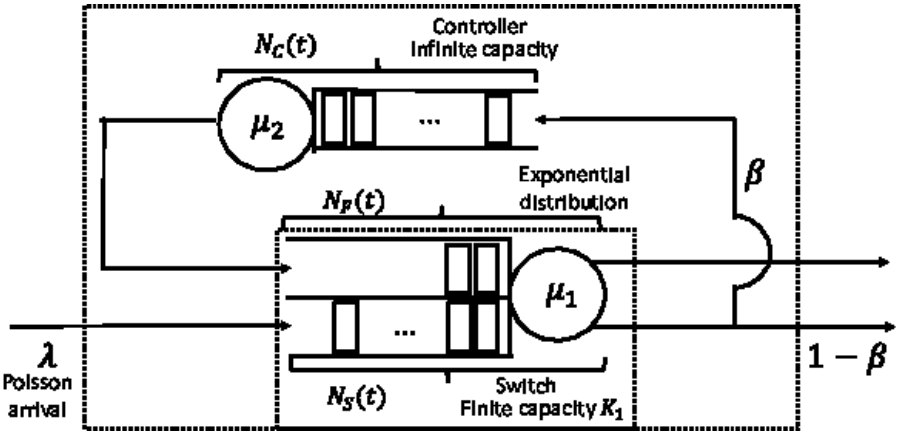


Figure 1.9: Queuing model of simple network with an OpenFlow switch connected to a single controller [23].

Figure 1.9 shows the OF network assumptions in [23], which consist of a single server and a switch. It is assumed that the incoming traffic is Poisson and it is called **class S**. The packets are looked up in the table; if there is a table miss, they are forwarded to the controller with probability (β) and labeled as **class C**. Otherwise, the packets are directed out of the network with probability (1 - β). When the packets are redirected from the controller to the switch they are labeled as **class F**, where they have higher priority than the newly arrived packets in the queue. The service times at the controller is exponentially distributed and the controller has infinite buffer size. The switch has a queue with finite size and geometrically distributed flow packet sizes. The authors calculate the number of jobs, the packet loss probability for each of the three classes,

and the average packet transfer delay in the system using a multi-dimensional Markov Chain model. This model is more accurate, because there is differentiation between the traffic from the controller and newly arriving traffic. Yet, the authors do not extend the work to multiple nodes. Moreover, they employ a priority queue to separate the packets into classes. The OF switches do not currently specify such separation.

The other strategy of analytical analysis is network calculus (NC), which considers the worst case performance bounds in the network such as throughput, delay in the system, and queue length [24]. The early performance studies of SDN network behaviour were studied by [25]. In that work, the authors quantified the packet delay and queue length bounds for the data plane and control plane. Their model represents a single node in the data plane. On the contrary, in [26] the authors have used calculus method to calculate the upper bound of the bandwidth, where latency is measured for a virtualized SDN controller. As reported in [27], the model of the system is based on stochastic network calculus with results obtained using Matlab. The authors proposed the lower bounds for the switch-controller end-to-end delay systems. In addition, their model is considered as the fastest tool used to measure the performance metric rather than using time consuming benchmarking tools.

The proposal studied in [28] measures the end-to-end delay in the network. The experimental results were obtained using switches of type OVS and controllers of POX type. Simulations were performed using OpenFlow on three different platforms; Mininet emulator, GENI, and OF@TEIN. The results show that M/G/1 models the end-to-end delay in OpenFlow enabled networks more accurately than M/M/1 models.

1.5 Main Contributions

The main contributions of this thesis are:

- Proposal of a class-based queueing network model to evaluate SDN network performance;
- Evaluation of the performance of the OpenFlow networks in different realistic scenarios;
- Extension of the existing analytical and simulation based studies to multiple controller and switch topologies;
- Use of a cost function that can be used for analysis of the performance and the energy consumption in the SDN;
- Presentation of a numerical analysis which justifies the effectiveness of having multi-controllers in scenarios with intermittent controller failures.

1.6 Thesis Structure

The remaining parts of this thesis are organized as follows: Chapter 2 presents the proposed model. Simulation results are discussed in Chapter 3. Conclusions and future work are given in Chapter 4.

Chapter 2

MODELS AND METHODS

2.1 Jackson Network Model of SDN

The Jackson queueing network is considered as the simplest model of packet routing in a computer network. In a Jackson queueing network, there are k nodes. In each node the service time is exponentially distributed with rate μ_i and there is an infinite buffer. In Jackson networks, each node can be viewed as an independent M/M/1 queue. Moreover, external arrivals to a node are considered as a Poisson process with rate r_i. The arrivals are served as first-come-first-served (FCFS). The routing of packets is probabilistic. Each packet finishing at node i will be sent to node j with probability P_{ij}, or will leave the network with likelihood $P_{i,out} = 1 - \sum_j P_{ij}$ [29].

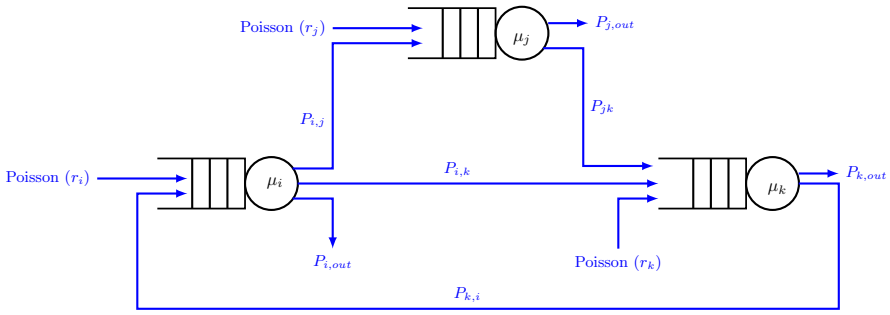


Figure 2.1: An example Jackson queueing network model [29].

Although it is now widely accepted that Poisson packet arrival and exponentially distributed packet size assumptions are not very realistic in modeling computer networks, these Markovian assumptions provide simple tractable analytical models

that enable designers to quickly understand how performance measures of interest vary as a function of certain parameters. This understanding is key to conducting more detailed empirical or experimental performance studies.

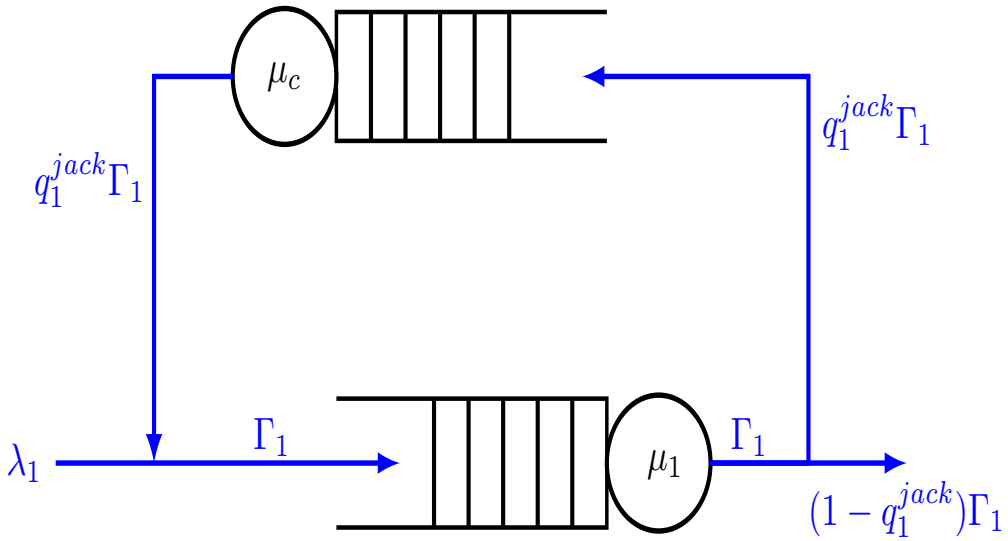


Figure 2.2: Jackson queueing network model applied to SDN [19]

Figure 2.2 shows the original configuration of Jackson network when applied to SDN which consists of one controller and one switch as described in [19]. The packets arrive at a switch with arrival rate λ_1. When a table miss happens in the flow table, the switch redirects the packet (or part of the packet) in the flow to the controller. This event is captured by probability q_1^{Jack}. Then, when the packet reaches to the controller, the controller finds the route for the packet installs the route into the switch, and the packet is redirected again to the switch. The net input to the controller Γ_c is equal to

$$\Gamma_c = q_1^{Jack}\Gamma_1, \tag{2.1}$$

where Γ_1 is the net input to the switch which is the summation of the controller output

and the outside traffic:

$$\Gamma_1 = \lambda_1 + q_1^{Jack}\Gamma_1. \tag{2.2}$$

The main shortcoming in Jackson network to model OpenFlow networks is that the processed packet (the packet which is returned from the controller) and the newly arrived packet from outside are treated identically by the switch. In other words, there is no differentiation between a newly arrived packet and a processed packet. In reality, traffic arriving at a switch that comes from the controller should not be redirected again to the controller; since the packet should now find a match in the flow table.

2.2 Modified Jackson Network

In order to avoid the shortcoming in Jackson network model in [19], the authors have modified the model of Jackson network in order to separate the traffic flow from the switch to the controller and vice versa correctly. The modified Jackson network is assumed to be a better representation of OF-based SDN network, and more accurately represents the OpenFlow operation rule. Figure 2.3 depicts the topology of the modified Jackson network.

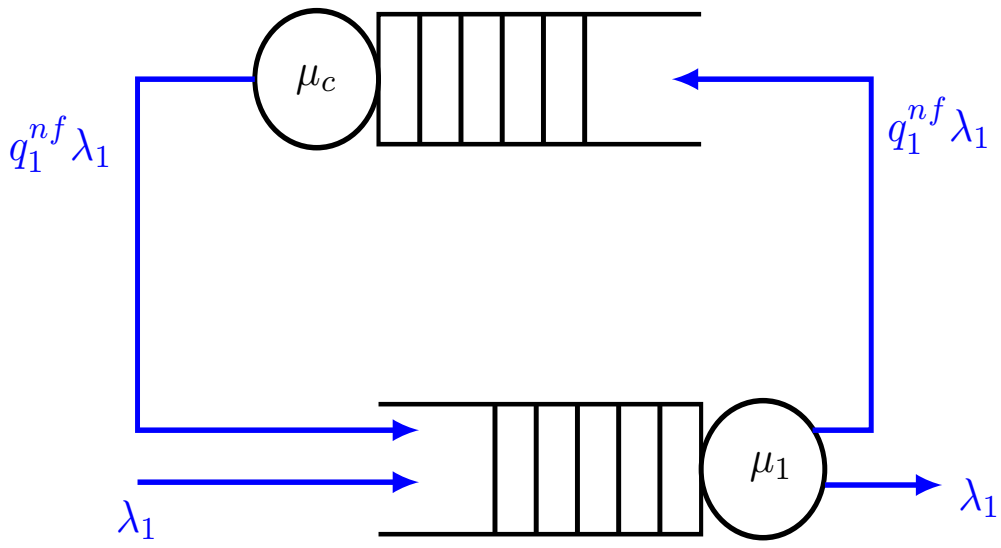


Figure 2.3: Modified Jackson model [19]

The net input to the controller Γ_c is now

$$\Gamma_c = q_1^{nf} \lambda_1, \tag{2.3}$$

where q_1^{nf} is the probability that the packet is forwarded to the controller when a table miss occurs. The probability q_1^{Jack} described in Section 2.1 needs some modification to model OF correctly when the net input to two models are the same. Therefore,

$$\Gamma_1 = \lambda_1 + q_1^{nf} \lambda_1, \tag{2.4}$$

and

$$q_1^{Jack} \Gamma_1 = q_1^{nf} \lambda_1, \tag{2.5}$$

so that

$$q_1^{Jack} = \frac{q_1^{nf}}{1 + q_1^{nf}}. \tag{2.6}$$

The authors in [19] thus propose an approximate solution to overcome the shortcoming of Jackson network by deriving the probability value when a table-miss occurs. Yet, by the nature of the Jackson networks, their model still fails to distinguish between the newly arrived packet and the processed packet by the controller. Moreover, their model is not easily to extendable to multiple node in control plane.

In the following sections, a model is proposed to overcome the difficulty in distinguishing between the newly arrived packets and the processed ones. A classed queuing network is suggested as an alternative model the SDN switch and the controller.

2.3 Classed Networks of Queues

In classed queueing networks there are k nodes and l classes of jobs (customers or packets). It is assumed that service time at each node is exponential with rate μ_i and an there is infinite buffer space. Hence, each node is viewed as an independent M/M/1 queue. Jobs at a node are served in FCFS manner. Each node may receive traffic from outside or inside the network. The outside traffic is a Poisson process with arrival rate r_i and follows a specific route depending on its class. The outside traffic can be from specific class (n). When the job is served by the node i, it will be transferred to node j with probability $P_{ij}^{(n)(n')}$, or it may leave the network $P_{i,out}^{(n)(n')}$. The lower indices in the notation $P_{ij}^{(n)(n')}$ refer to sending the jobs from the node i to node j, while the upper indices in that notation refer to a customer class change from class n to class n'. In this type of networks, jobs can change the class, visit the node multiple times, and each class can have different priorities [29]. Figure 2.4 shows a simple class based queuing network.

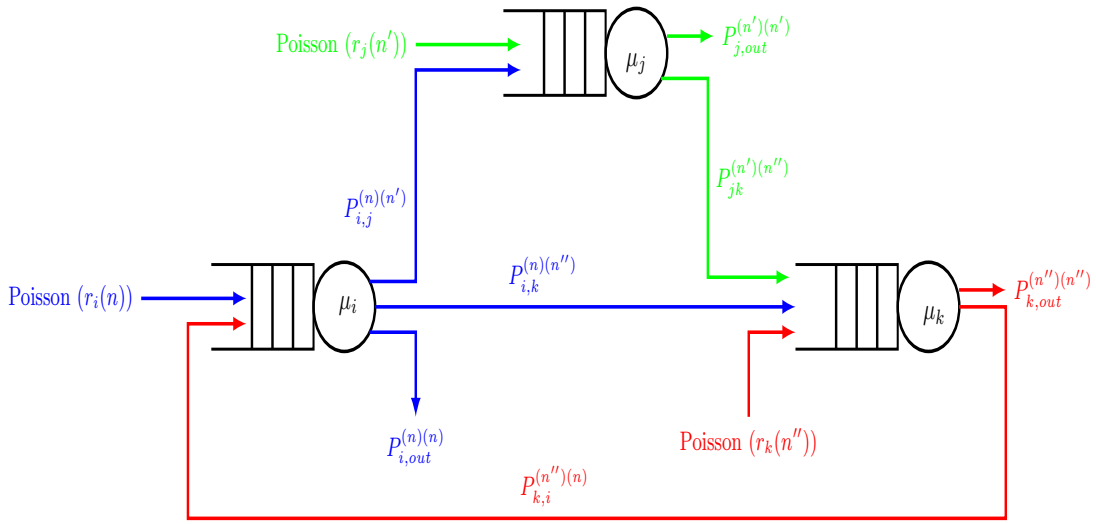


Figure 2.4: Simple classed network of queues

2.4 Proposed Model for SDN Using Classed Networks of Queues

The proposed model is derived based on classed queueing networks where different classes are assigned to the processed packet and the newly arrived packet. In the following discussion, the packets that visit the controller will be designated as class 0 packets. Thus, the switch can discriminate between different types of packets based on their class. How packet classes in the presented methodology are changed is illustrated in Figure 2.5. Three different topologies are analyzed. First, we study a simple topology which consists of one controller and one switch. Second, two more switches are added (in series with the first one) to the data plane. Finally, two controllers and one switch are used to build the network, where results are obtained using different methods of packet dispatching.

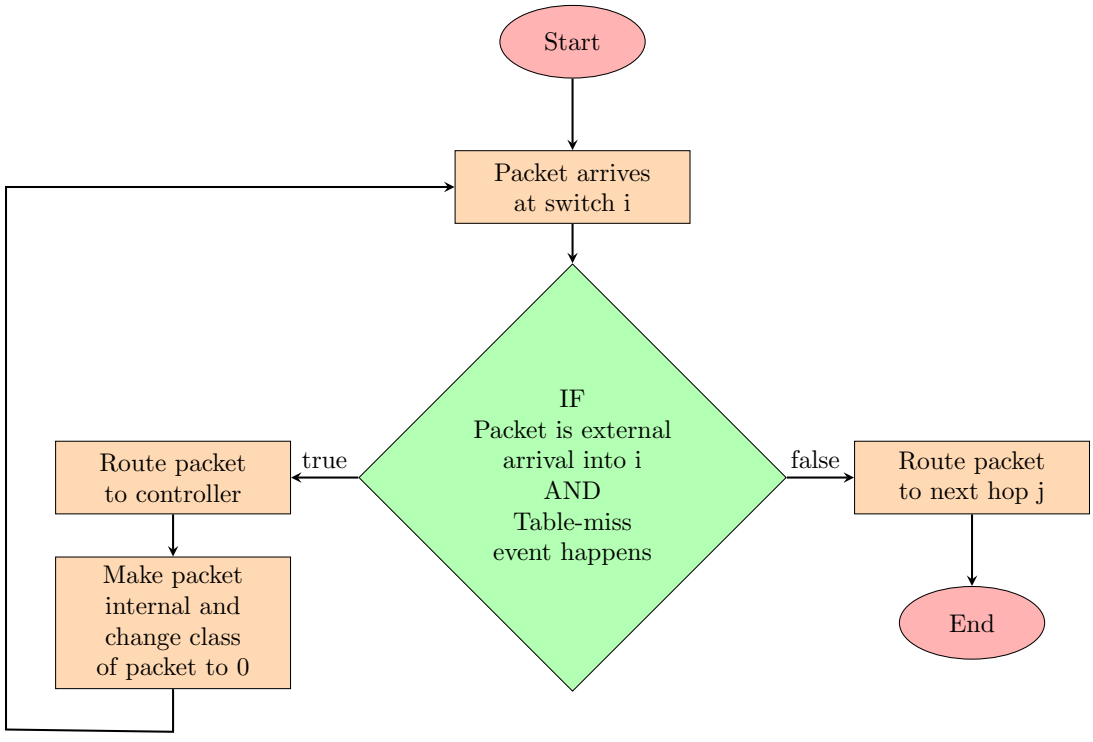


Figure 2.5: Flowchart of proposed model

2.4.1 One Controller, One Switch Topology

Figure 2.6 shows a simple SDN topology which consists of one controller and one switch. The external traffic arrival rate denoted by $r_1(1)$ reaches to the switch; this traffic belongs to **class (1)**. If there is no flow entry in the flow table for a packet, the packet is encapsulated in a "PACKETIN" message, which is transferred to the controller. This event is supposed to happen with a probability $P_{1,c}^{(1)(1)}$. The lower indices in the notation $P_{1,c}^{(1)(1)}$ refer to the probability of sending the packet from the switch (1) to the controller (c), while the upper indices in that notation refer to no class change. This notation is applicable for all the following topologies. It is assumed that after the controller updates the flow table, the packet class is changed from **class (1)** to **class (0)** and it is transmitted again to the switch with probability $P_{c,1}^{(1)(0)}$. Whenever class (0) reaches the switch, it is directed outside the network with probability $P_{1,out}^{(0)(0)}$.

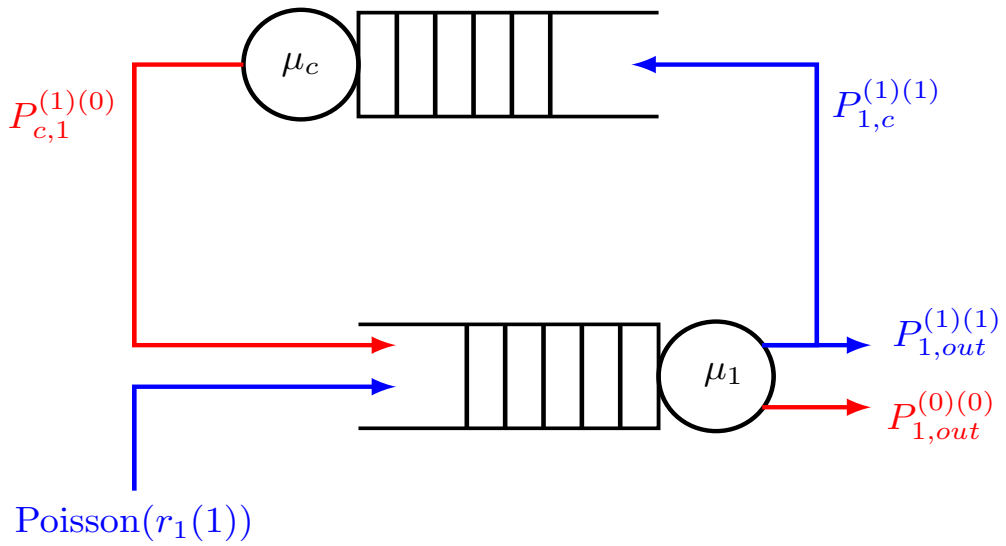


Figure 2.6: Simple OF network of proposed model

If there is a match in the flow table, the switch applies the actions and directs the packet outside the network with probability $P_{1,out}^{(1)(1)} = 1 - P_{1,c}^{(1)(1)}$, where the lower index in $P_{1,out}^{(1)(1)}$ refers to the packet class (1) is leaving the network (out), and higher index

refers to no-change status in the class.

2.4.2 Extended Data Plane Topology

The data plane can be extended using multiple switches and one controller to build the network. An example network configuration used in simulation is the network shown in Figure 2.7. In this network, three switches are connected in series, labeled as switch 1, switch 2 and switch 3 with external arrivals tagged as class (1), class (2) and class (3), respectively. The packets that visit the controller become class 0. The communications between the controller and the switches are as follows: when a packet reaches the switch with no predefined entry in the flow table, a "PACKETIN" message is produced by the switch and it is sent to the controller. The controller determines the forwarding path and distributes it to all the switches on the path in order to add an entry in the flow tables. Therefore, if the flow has an entry on the table it is forwarded directly to the next switch on the path.

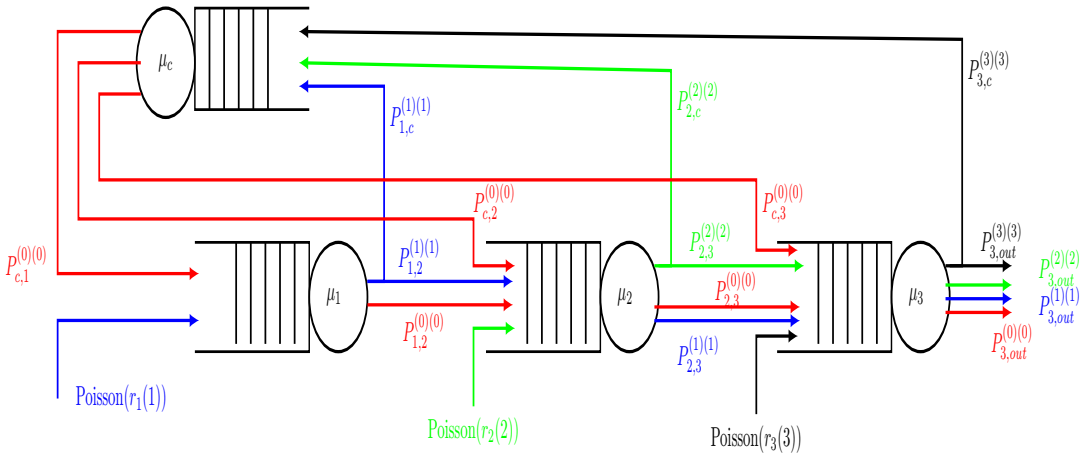


Figure 2.7: Extended data plane

If there is an entry for the class (1) flow, it is forwarded from "switch 1", to "switch 2", "switch 3", and then outside the network. Therefore, when class (1) flow is received by switch 1, where there is in no entry in the flow table, then it will be directed to

the controller; it is assumed that this event happens with probability $P_{1,c}^{(1)(1)}$. Then the controller computes the path and a new entry is added to the flow tables in the switches. Furthermore, the controller changes the packet class from class (1) to class (0), where class (0) is treated as there is an entry matching and it is directed form switch 1 to 2 and 3. This discussion is under the assumption that, if there is a packet matching at switch 1, then there will be a match at switches 2 and 3. Notice that if the flow has different class (say class (2) or class (3)) then it will be treated similar to the scenario of class (1) discussed above.

2.4.3 Topology with Multiple Controller

In single controller topology, there is a single point of failure. If failure happens in the controller, the switches lose the communication with the controller and the network will be down. As the network traffic increases, one controller cannot process huge flow requests received from the switches due to its limited capacity. Thus, multiple controller topology is suggested for real networks like data centers, and cloud networking to improve the network reliability. Multiple controllers are better in network load management. Also, the traffic load can be shared between the several controllers, where the controllers keep consistent information in order to transmit packets correctly. Moreover, multiple controller topology has redundancy which reduces the chance of network failure. It is less likely to have two controllers failing at the same time. Thus, the second controller carries on and controls the network. In this respect, the multiple controller topology needs to be analyzed as well.

The network can be supplied with homogeneous controllers, where all controllers have the same specifications in terms of buffer size, memory, and service rate [30]. Furthermore, the network may be supplied with heterogeneous controllers, where the

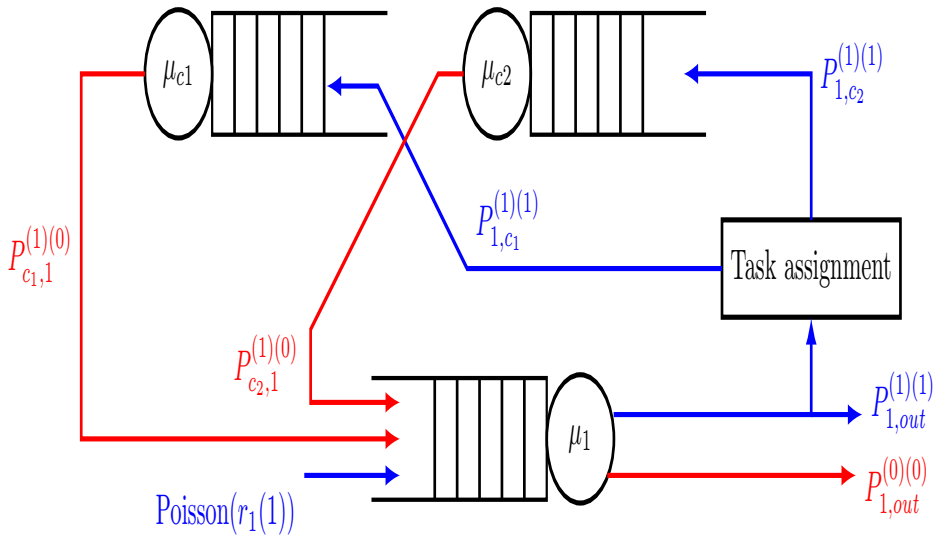


Figure 2.8: Extended control plane

controllers have different specifications as in [31]. Figure 2.8 shows the topology of multiple homogeneous controllers. The external traffic reaches the switch with rate $r_1(1)$ and is referred to as **class(1)**. If there is no match in the flow table, a "PACKETIN" message is created and sent to one of the controllers. This event is assumed to occur with probability $P^{(1)(1)}_{1,c_i}$, where $P^{(1)(1)}_{1,c_i}$ is the probability that the packet is directed to the controller when there is no match in the flow table. When the packet matches an entry, it directly applies the action field with probability $(1-P^{(1)(1)}_{1,c_i})$. In order to choose one of controllers efficiently, we assumed that there is a routing component or task allocator that chooses the controller, where a special policy is used to determine to which controller the switch should interact. This policy is called task assignment policy, and also known as load balancer or dispatcher [29]. The controllers are assumed to imeediately feedback their status to the task allocator.

There are different ways to dispatch packets to the controllers: random, round robin (RR), and join the shortest queue (JSQ). In the first type, the packet is directed to one of the controllers randomly with equally likely probability. The aim of this policy is

to balance the number of packets at each controller. On the other hand, round robin strategy distributes the packet from the switch to the controllers in a round robin form. JSQ policy is also called shortest queue length, the packet joins the controller which has the least number of packets in the queue.

2.4.4 Energy Performance Trade-off

In multiple controller topologies, a central problem is how to use efficient number of controllers to extract the best performance under unpredictable traffic while not wasting energy. In order to study the trade-off between the performance and the energy consumed by the controller, we introduce a cost function to capture this trade-off, where the energy consumed by the controller depends on its processing speed. At low processing speeds, the performance of the system decreases, while at higher speeds, more energy is consumed.

Based on [32, 33], the minimum power (P) needed by the processor is given by

$$P \propto \mu_c^{\alpha} \tag{2.7}$$

where μ_c is the service rate of the controller in (packets/second) and α is a parameter which is typically chosen as 2 or 3. For digital electronic devices, the power-delay product (PDP), which is the product of power consumption P and input-output delay. Similarly, the concept of PDP is applied here for the network system, where the input-output delay is the system response time (R). Thus, a cost function F is defined as

$$F = \gamma \times P \times R. \tag{2.8}$$

The constant γ is a function of α and is $\gamma = \frac{\text{sec}^{\alpha-1}}{\text{pkt}^{\alpha}}$. When $\alpha = 2$ the cost function is

given for one controller topology as

$$F = \mu_c^2 \times R \times \gamma. \tag{2.9}$$

For multiple controllers, the service rate for each controller is taken as the half of that of the controller in one controller topology. It is important to point out that the total power consumed by multiple controllers is found as the sum of the individual controller. From (2.7), one may observe that the power of the controller depends on the service rate. Thus, in case of two controllers, total power is given as $(\frac{\mu_c}{2})^2 + (\frac{\mu_c}{2})^2 = \frac{\mu_c^2}{2}$. As a results, the cost function in case of double controllers is

$$F = \frac{\mu_c^2}{2} \times R^{'} \times \gamma, \tag{2.10}$$

where $R^{'}$ is the system response time in case of multiple controller.

Similarly, for the case when $\alpha = 3$, the cost function of one controller is given as

$$F = \mu_c^3 \times R \times \gamma, \tag{2.11}$$

and for multiple controller is given as

$$\begin{aligned} F =& \left\{ (\frac{\mu_c}{2})^3 + (\frac{\mu_c}{2})^3 \right\} \times R^{'} \times \gamma \\ =& \frac{\mu_c^3}{4} \times R^{'} \times \gamma. \end{aligned} \tag{2.12}$$

Chapter 3

RESULTS AND DISCUSSION

In order to verify our models, we simulated our topologies using Java Modeling Tool (JMT) version V.1.0.2. JMT is a discrete event simulator which was developed at Politecnico di Milano and Imperial College London to evaluate the performance of queueing networks [34]. The simulations were executed using a TOSHIBA laptop with an Intel Core i7 processor. Each simulation runs until a 95% confidence is obtained for the performance measure of interest with less than 3% maximum relative error or the simulation has analyzed 1,000,000 samples. These stopping conditions ensure that the performance indices of interest are reported with high precision as long as the simulation time is not prohibitive. The simulations were executed considering the following assumptions:

1. The controllers and the switches have infinite queues (assuming no packets losses) [18, 19].

2. The service times at the controllers follow an exponential distribution with rate μ_c and mean value of 240 μs [18, 19].

3. The service times at the switches follow an exponential distribution with rate μ_1 and mean value of 9.8 μs [18, 19].

4. All of the external traffic arriving into the network is assumed to be modeled as a Poisson process.

3.1 One-Controller, One-Switch Topology

The performance of the network shown in Figure 2.6 is given in Figure 3.1. The figure shows the relation between the system sojourn time and the controller utilization ρ_c, where the sojourn time is defined as the average time spent by the packet in the network from the moment the packet enters the network till it leaves it.

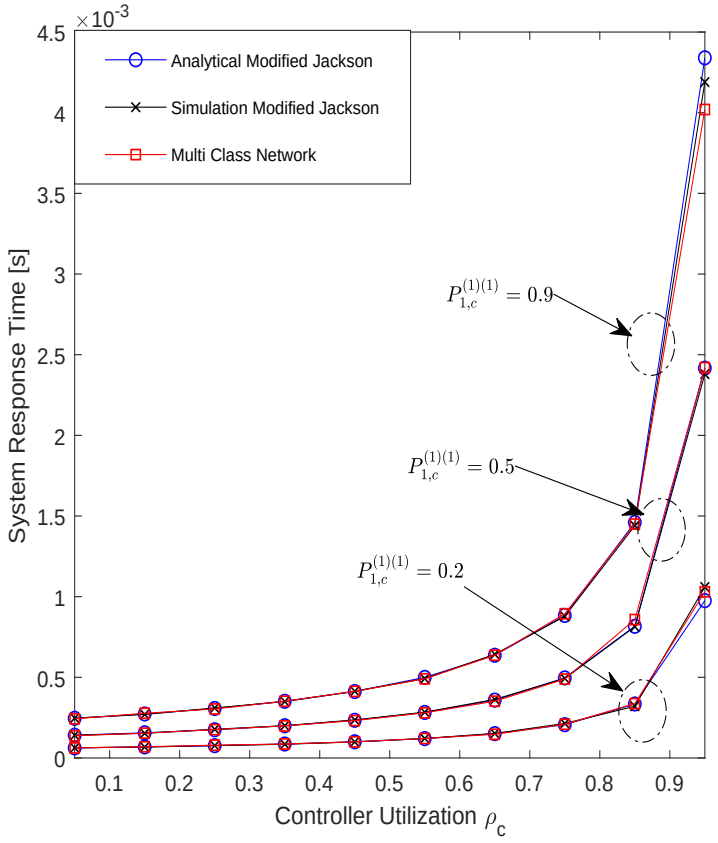


Figure 3.1: Comparison of the proposed model to the modified Jackson model

The authors in [19] validate their model using OMNeT++ packet-level discrete event simulation results reported in [18]. The results obtained in this thesis are compared with the results obtained in [19] and thus with the simulations in [18]. The analytical modified Jackson and simulation modified Jackson in Figure 3.1 refers to the models described in Section 2.2 which were validated with packet-level simulations in [18]. The multiclass network in Figure 3.1 refers to the simulation results obtained from the

proposed model in Section 2.4.1 of this thesis.

The simulations were performed under different probability values $P_{1,C}^{(1)(1)}$ for traffic being directed to the controller. The range of external arrival rate is 2083.3 to 18750 pkt/s for $P_{1,C}^{(1)(1)} = .2$, 833.33 to 7500 pkt/s for $P_{1,C}^{(1)(1)} = .5$, and 462.96 to 4166.7 pkt/s for $P_{1,C}^{(1)(1)} = .9$. It can be seen that the system response time grows exponentially with increasing controller utilization. Also, when a single controller is used in the network, the performance results of the proposed topology coincides with the analytical and simulation results of modified Jackson model.

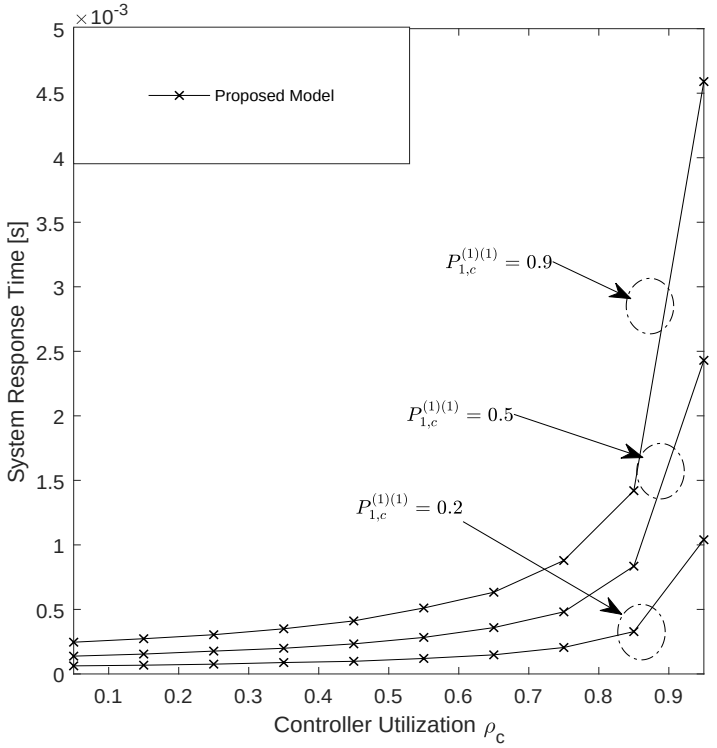


Figure 3.2: Extended data plane with three switches

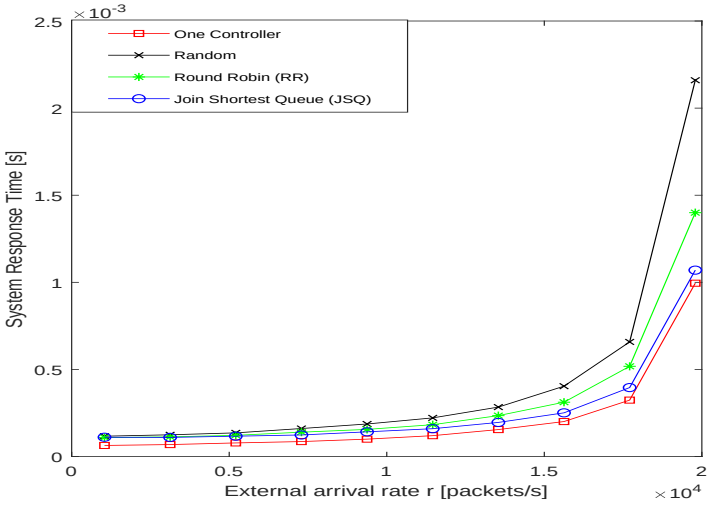
3.2 Extended Data Plane

Figure 3.2 shows the simulation results for extended data plane where there are three switches. In the simulations, it is assumed that the external arrival rate is equal for all switches (i.e: $r_1(1)$=$r_2(2)$=$r_3(3)$). The range of external arrival rate is 694.43 to

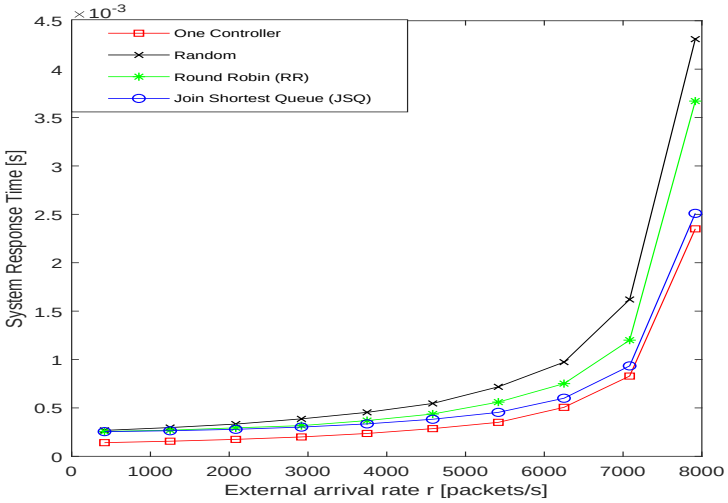
6250 pkt/s for $P_{1,C}^{(1)(1)} = 0.2$, 277.7 to 2500 pkt/s for $P_{1,C}^{(1)(1)} = 0.5$, and 154.32 to 1388.9 pkt/s for $P_{1,C}^{(1)(1)} = 0.9$. Simulations were performed at different probability values of table miss occurrence in order to investigate the proposed topology. The probability of table miss in all switches is assumed to be the same $P_{1,C}^{(1)(1)}$=$P_{2,C}^{(2)(2)}$=$P_{3,C}^{(3)(3)}$. Also the service rate at each switch is the same μ_1=μ_2=$\mu_3 = 9.8\mu$s. Comparing Figures 3.1 and 3.2, one can see that the response times with one switch and three switches are similar when the total external arrival rates are the same. This can be attributed to the fact that the controller is the bottleneck and is thus the limiting factor of performance.

3.3 Multiple Controller Topology

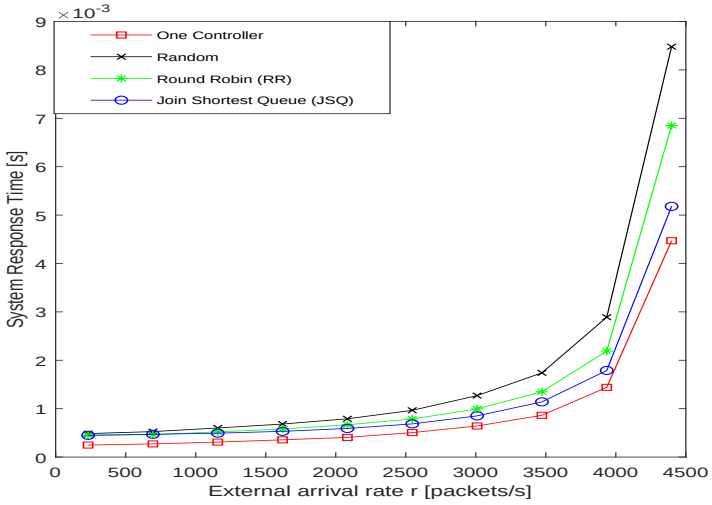
In case we have one switch and two controllers as discussed in Section 2.4.3, each controller has half service rate ($\frac{\mu_c}{2}$) compared to the scenarios studied in Section 2.4.1 and Section 2.4.2 for fair comparison. In multiple controller topology, it is required to use either random, round robin or JSQ dispatchers in order to dispatch the packets from the switch to the controller. A comparison is given between the three methods in order to show the most effective method to be used in the multi-controller topology. The comparison is done for three different probabilities of traffic going to the controller ($P_{1,c_i} = 0.2, P_{1,c_i} = 0.5$ and $P_{1,c_i} = 0.9$) as shown in Figure 3.3. It is clear from the figure that the JSQ strategy has the fastest system response time compared to the round robin and random strategies. The response times of the one-controller topology is provided in Figure 3.3 in order to give a lower bound for the comparison, where the one-controller topology has the fastest response time compared to multi-controllers approaches. An important observation is that the response time for the random strategy is twice as much the response time with one controller case. The results obtained for average response times with two controllers are in agreement with the comparison of different task assignment policies given in [29]. Since round-robin results in less variability in



(a) Probability of table miss $P_{1,C}^{(1)(1)} = 0.2$.



(b) Probability of table miss $P_{1,C}^{(1)(1)} = 0.5$



(c) Probability of table miss $P_{1,C}^{(1)(1)} = 0.9$

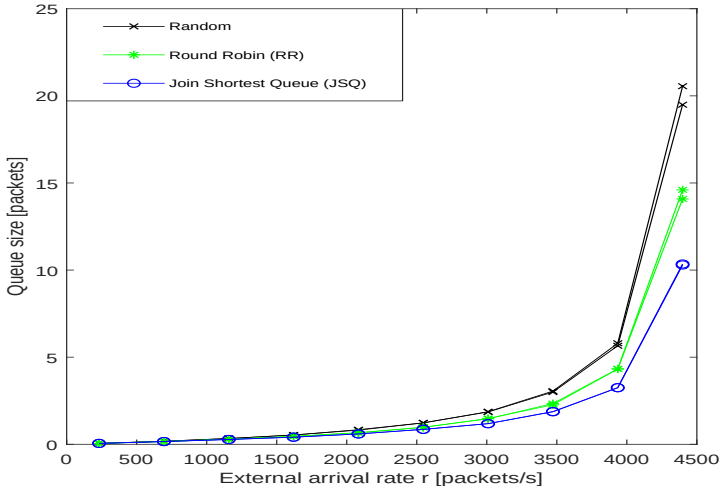
Figure 3.3: Comparison of different dispatching policies

inter arrival times at controllers compared to random task assignment, it is expected to outperform random task assignment. JSQ, on the other hand, is a dynamic policy; it attempts to equalize the instantaneous number of packets at each queue and reacts quickly to changes. Hence, JSQ outperforms round-robin which tries to equalize the average number of packets. Figure 3.4 further depicts the queue size for the three task assignment policies which result in similar findings and trends as those obtained for response times.

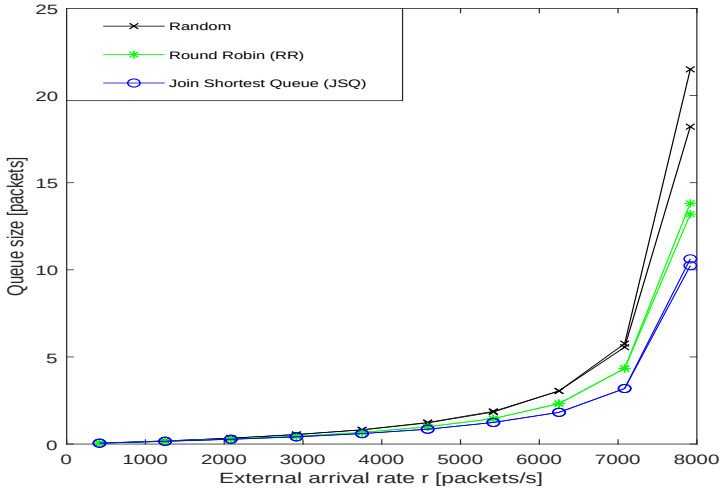
3.3.1 Energy Performance Trade-off

In the following, the cost function (2.8) is evaluated using simulation for a wide range of arrival rates at different probabilities of table miss. At each probability value, the one-controller topology, random, RR and JSQ algorithms are compared, where the one-controller topology can be seen to give an upper bound for the cost function value compared to multiple controllers.

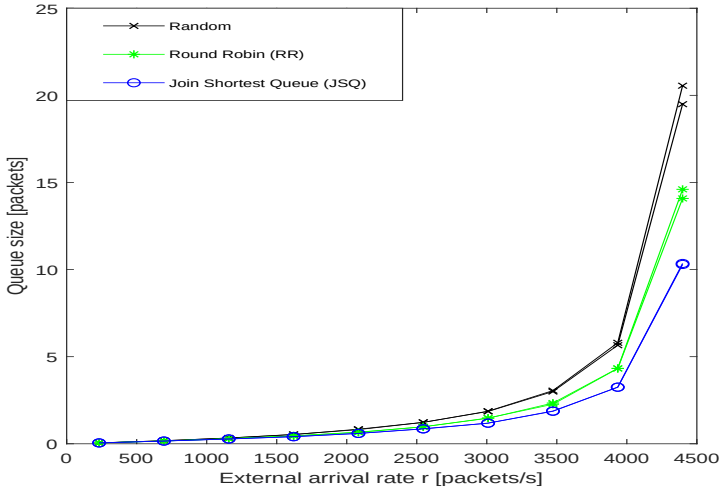
The analyses were conducted for the cases when $\alpha = 2$ and $\alpha = 3$ with $\gamma = 1$. Figure 3.5 depicts the cost function (2.10) when $\alpha = 2$ for two controllers and (2.9) for one controller. The results clearly show that the cost function for one controller and random dispatcher coincide for different probabilities of table-miss considered, and random strategy is the highest cost compared to RR and JSQ, whereas the JSQ is the least cost strategy. Figure 3.6 shows the cost function (2.11) when $\alpha = 3$. The figure shows that the cost for one controller topology is twice higher than the random strategy for all different probabilities. While the JSQ has the least cost function values compared to the other policies. One may conclude that using multiple controllers is an effective strategy from energy efficiency point of view compared to single controller. This advantage promotes the multiple controller topology to be used in the market by the designers.



(a) Probability of table miss $P_{1,C}^{(1)(1)} = 0.2$.

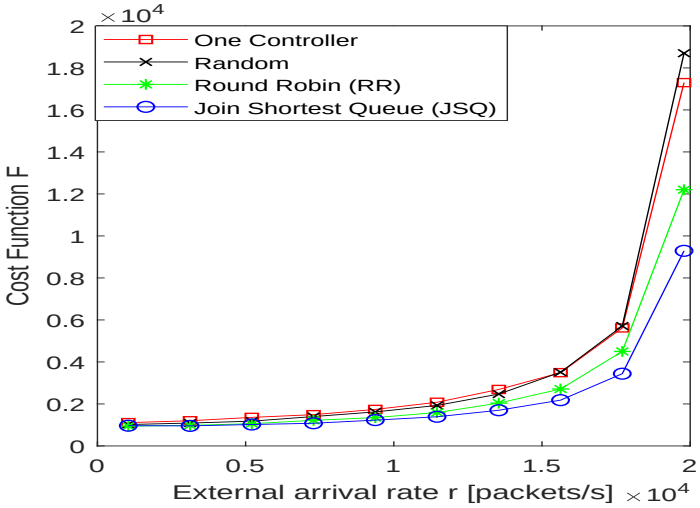


(b) Probability of table miss $P_{1,C}^{(1)(1)} = 0.5$

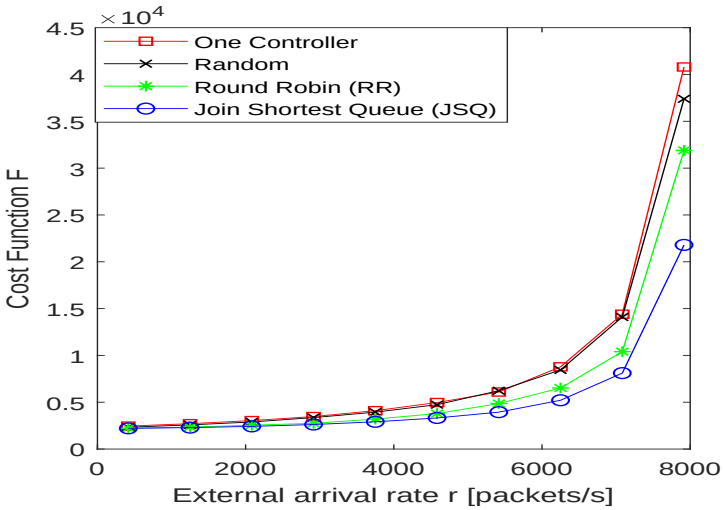


(c) Probability of table miss $P_{1,C}^{(1)(1)} = 0.9$.

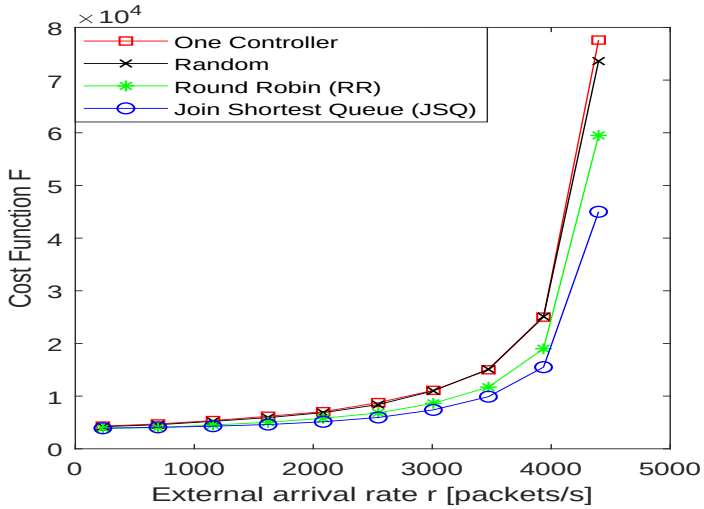
Figure 3.4: Queue sizes for each of the two controllers under different dispatching policies



(a) Probability of table miss $P_{1,C}^{(1)(1)} = 0.2$.

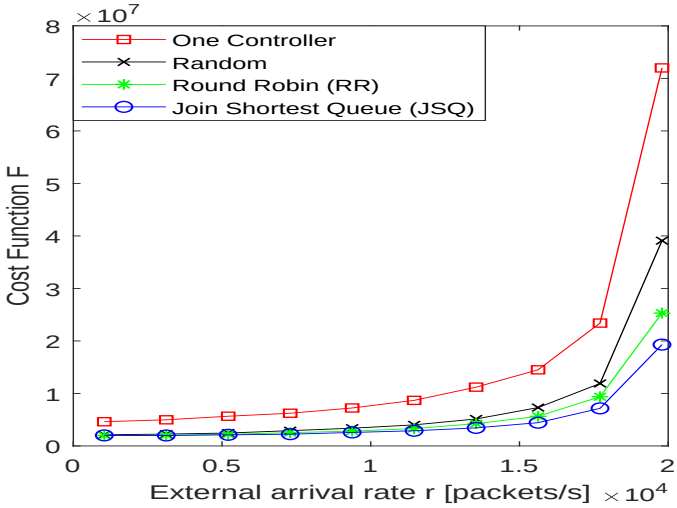


(b) Probability of table miss $P_{1,C}^{(1)(1)} = 0.5$

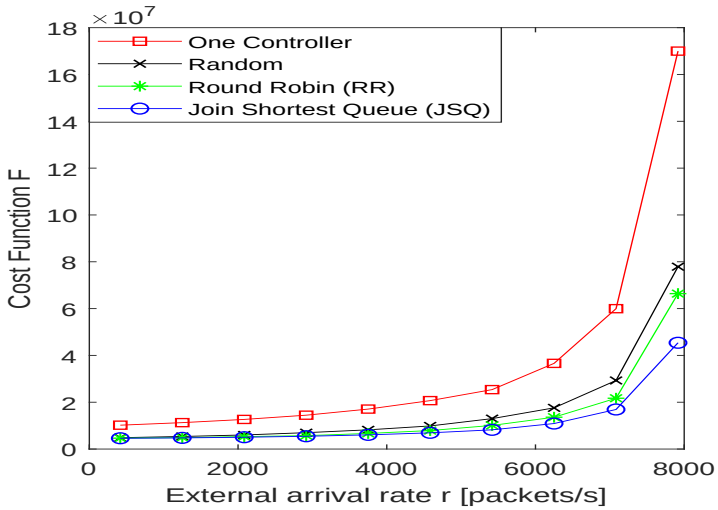


(c) Probability of table miss $P_{1,C}^{(1)(1)} = 0.9$

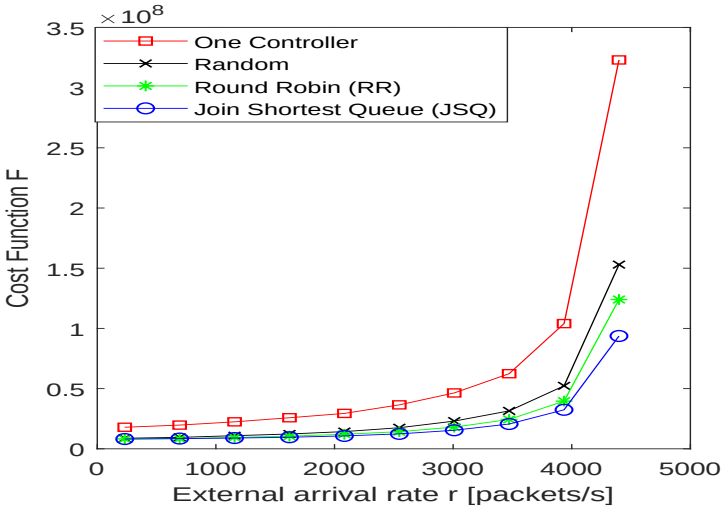
Figure 3.5: Cost function when $\alpha = 2$



(a) Probability of table miss $P_{1,C}^{(1)(1)} = 0.2$

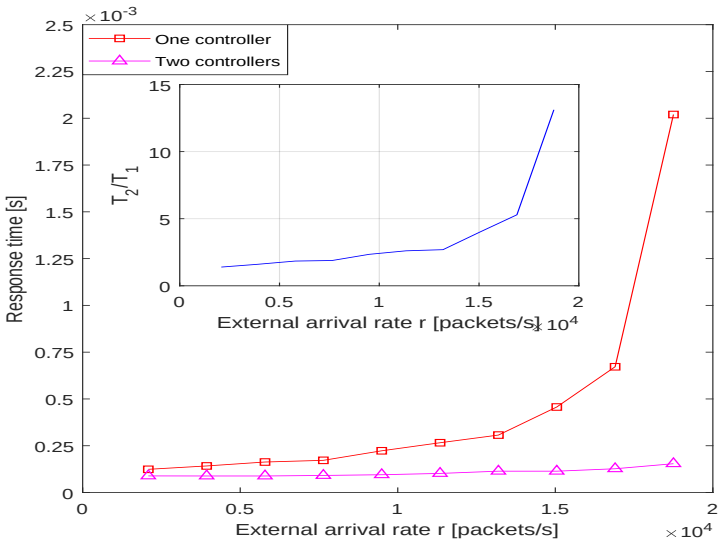


(b) Probability of table miss $P_{1,C}^{(1)(1)} = 0.5$

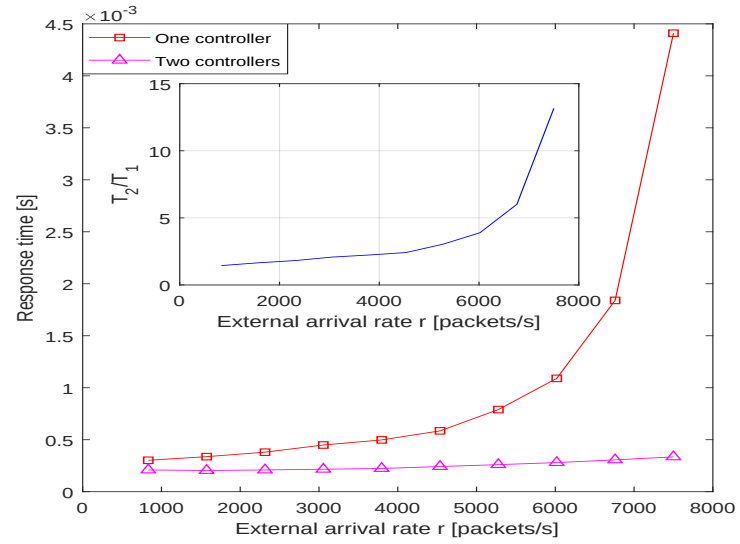


(c) Probability of table miss $P_{1,C}^{(1)(1)} = 0.9$

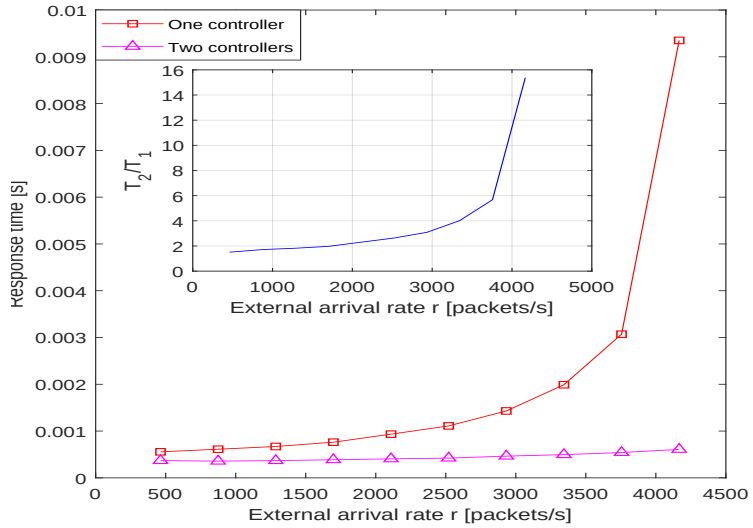
Figure 3.6: Cost function when $\alpha = 3$



(a) Probability of table miss $P_{1,C}^{(1)(1)} = 0.2$.



(b) Probability of table miss $P_{1,C}^{(1)(1)} = 0.5$



(c) Probability of table miss $P_{1,C}^{(1)(1)} = 0.9$

Figure 3.7: Controllers with intermittent failures

3.3.2 Effect of Redundancy with Intermittently Failing Controllers

Use of multiple SDN controllers may be necessary to prevent service outages and delays when controllers experience intermittent failures. Increasing the number of controllers however will increase capital and operational costs. Therefore, such increases can be compensated by the corresponding reductions in packet delays. A quantitative analysis of the decrease in delays when the number of controllers is increased is necessary. In this work, the focus will be on using two controllers.

Service outages in SDN controllers can be due to hardware, software, or link failures. Although recent research tries to characterize the nature of failure dynamics in SDN controllers [35, 36], there are still no well-established failure models. As such, a relatively simple failure model of controllers is considered in this thesis. It is assumed that the controllers intermittently fail. The failures are simulated by using the following service time model: The time at controllers is assumed to be slotted and each slot is 0.01 seconds. In each slot, the controllers fail with a probability of 0.05; this failure case is simulated by choosing service times from an exponential distribution with a mean of 0.24 seconds. The controllers work normally with a probability of 0.95; here, the service times are chosen from an exponential distribution with a mean of 240 microseconds. When there are two controllers, it is also assumed that there is a routing component or task allocator that chooses the controller with the fastest service (or the one that is not failing); this information is supposed to be fed back to the task allocator.

Figure 3.7 shows that with only one controller, the packet response times grow exponentially with increasing arrival rates. The packet response times are greatly improved if two controllers are employed, especially at high arrival rates. The

throughput of packets can be defined as the inverse of response times (with units of packets per second). Denoting the throughput with one controller as T_1 and with two controllers as T_2, one can calculate the ratio T_2/T_1 to illustrate the throughput gain when two controllers are employed. The insets of the figures show T_2/T_1. Note that one can find that the gain exceeds 2 when the external arrival rates are roughly larger than 10,000, 3,500, and 2,000 packets/second when p_c is 0.2, 0.5, and 0.9, respectively; these all correspond to a controller utilization of approximately 50% in the one controller scenario. Finally, Figure 3.8 compares all the cases considered in this section. It can be seen that P_{1,c_i} greatly affects the response times for a given arrival rate.

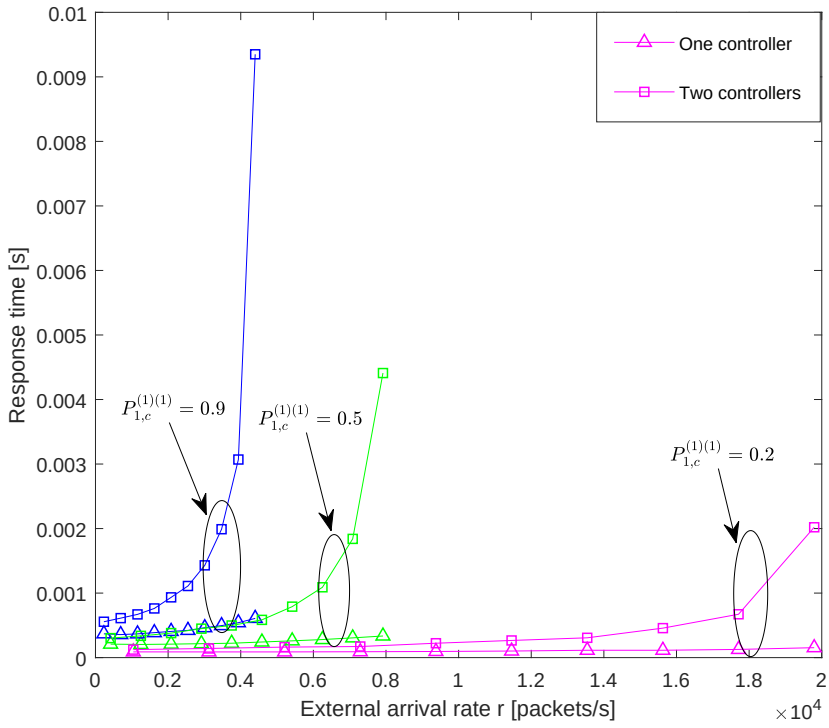


Figure 3.8: Failure all scenarios

3.3.3 Benefits of Employing Multiple Controllers

Simulation results shown in Figures 3.5 and 3.6 and the statistical details provided in the Appendix indicate that the power-delay performance is improved when two

half-capacity controllers are deployed instead of a single full-capacity controller. In addition, in case there are intermittent controller failures, installing two controllers may be justified for large traffic loads as illustrated in Figure 3.7 with statistical details reported in the Appendix.

Chapter 4

CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

In this thesis, the SDN networks is modeled using classed queueing network. The proposed network model is considered as a simple model which can be generalized to multiple switch and controller topologies. The basic concept of classes is explained and it is used as the core idea of the proposed model. The key idea behind the proposed model is the usage of classes, where the packet is directed to the controller if there is table-miss in the switch. The controller changes the packet's class and forwards it to the switch again. Thus, the switch distinguishes between the newly arriving packets and the packets which are coming from the controller.

The proposed model is used to model three different topologies, namely, single controller, extended data plane and multiple controllers topology. The presented work is compared with previous work and gives very close results in the sojourn time of the packets when there is one controller and one switch. By adding two additional switches, we extend the data plane, simulating a more realistic SDN scenario.

It is shown that multiple controller topology is more effective and powerful than one controller topology. The use of multi-controllers increases the network's redundancy in case of controller failures, compared to one controller topology where the system completely fails. Additionally, a cost function is employed to analyze the system for

energy consumption and best performance. The results show that the multiple controller topology has less cost in terms of power delay product when compared to a single controller topology.

4.2 Future Work

By looking to the future, we are looking forward to validating the proposed model experimentally. Indeed, the experimental validation will show the network limitations in terms of hardware validation and other constraints which may arise in practice. Note, however, that experimental validation of the models can prove difficult since SDN is not very widely implemented yet. There is currently only limited research work that use actual SDN measurement data for performance analysis. Moreover, the work presented here is based on Markovian arrival assumption. As a future work, one may extend the application of the proposed model so that different arrival distributions which are more suitable for modeling computer network traffic are employed.

REFERENCES

[1] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, July 2013.

[2] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. Van Reijendam, P. Weissmann, and N. Mckeown, "Maturing of OpenFlow and Software-Defined Networking through deployments," *Computer Networks*, vol. 61, pp. 151–175, 2014.

[3] K. Greene, "MIT Tech Review 10 Breakthrough Technologies: Software-Defined Networking," http://www2.technologyreview.com/article/412194/tr10-software-defined-networking/, 2009.

[4] S. Shenker, M. Casado, T. Koponen, N. McKeown *et al.*, "The future of networking, and the past of protocols," *Open Networking Summit*, vol. 20, pp. 1–30, 2011.

[5] N. McKeown, "How SDN will shape networking," *Open Networking Summit*, 2011.

[6] 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.

[7] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[8] W. Stallings, *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 2015.

[9] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Computer Networks*, vol. 71, pp. 1–30, 2014.

[10] D. Hoang, "Software defined networking shaping up for the next disruptive step," https://telsoc.org/ajtde/2015-12-v3-n4/a28, 2019, accessed: 2019-5-5.

[11] P. Chhikara, "Centalized/distributed control," https://www.slideshare.net/PallaviChhikara/sdn-ppt, accessed: 2018-11-11.

[12] 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.

[13] M. HOWARD, "Open networking foundation,"

https://www.opennetworking.org/, accessed: 2018-9-11.

[14] O. S. Specification, "Version 1.0. 0 (Wire Protocol 0x01)," *Open Networking Foundation*, 2009.

[15] C. Ching-Hao and Y.-D. Lin, "Openflow version roadmap," 2015.

[16] P. Goransson, C. Black, and T. Culver, *Software defined networks: a comprehensive approach*. Morgan Kaufmann, 2016.

[17] L. Kleinrock, *Queueing systems, volume 2: Computer applications*. wiley New York, 1976, vol. 66.

[18] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," in *Proceedings of the 23rd International Teletraffic Congress*. International Teletraffic Congress, 2011, pp. 1–7.

[19] K. Mahmood, A. Chilwan, O. N. Østerbø, and M. Jarschel, "On the modeling of OpenFlow-based SDNs: The single node case," *Proceedings of Computer Science and Information Technology (CS & IT)*, vol. 4, pp. 207–217, 2014.

[20] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, "Performance evaluation of OpenFlow-based software-defined networks based on queueing model," *Computer Networks*, vol. 102, pp. 172–185, 2016.

[21] B. Xiong, X. Peng, and J. Zhao, "A Concise Queuing Model for Controller Performance in Software-Defined Networks." *JCP*, vol. 11, no. 3, pp. 232–237, 2016.

[22] K. Sood, S. Yu, and Y. Xiang, "Performance analysis of software-defined network switch using M/Geo/1 model," *IEEE Communications Letters*, vol. 20, no. 12, pp. 2522–2525, 2016.

[23] Y. Goto, H. Masuyama, B. Ng, W. K. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic OpenFlow–based switch model," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016 IEEE 24th International Symposium on*. IEEE, 2016, pp. 301–306.

[24] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer Science & Business Media, 2001, vol. 2050.

[25] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou, "An analytical model for software defined networking: A network calculus-based approach," in *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 1397–1402.

[26] A. G. Osgouei, A. K. Koohanestani, H. Saidi, and A. Fanian, "Analytical performance model of virtualized SDNs using network calculus," in *Electrical Engineering (ICEE), 2015 23rd Iranian Conference on*. IEEE, 2015, pp. 770–

774.

[27] C. Lin, C. Wu, M. Huang, Z. Wen, and Q. Zheng, "Performance evaluation for SDN deployment: An approach based on stochastic network calculus," *China Communications*, vol. 13, no. Supplement 1, pp. 98–106, 2016.

[28] A. Iqbal, U. Javed, S. Saleh, J. Kim, J. S. Alowibdi, and M. U. Ilyas, "Analytical modeling of end-to-end delay in openflow based networks," *IEEE Access*, vol. 5, pp. 6859–6871, 2017.

[29] M. Harchol-Balter, *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[30] Y. Jimenez, C. Cervello-Pastor, and A. J. Garcia, "On the controller placement for designing a distributed SDN control layer," in *Networking Conference, 2014 IFIP*. IEEE, 2014, pp. 1–9.

[31] M. Mendonca, K. Obraczka, and T. Turletti, "The case for software-defined networking in heterogeneous networked environments," in *Proceedings of the 2012 ACM conference on CoNEXT student workshop*. ACM, 2012, pp. 59–60.

[32] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*. IEEE, 1995, pp. 374–382.

[33] M. Andrews, S. Antonakopoulos, and L. Zhang, "Energy-aware scheduling

algorithms for network stability," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1359–1367.

[34] M. Bertoli, G. Casale, and G. Serazzi, "JMT: performance engineering tools for system modeling," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 10–15, 2009.

[35] T. A. Nguyen, T. Eom, S. An, J. S. Park, J. B. Hong, and D. S. Kim, "Availability modeling and analysis for software defined networks," in *2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2015, pp. 159–168.

[36] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, and C. M. Machuca, "Characterization of failure dynamics in sdn controllers," in *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*. IEEE, 2017, pp. 1–7.

APPENDIX

Statistical Details of Results for Multiple Controller Topologies

The values in the shaded cells were not computed with the required precision of 95% confidence level and 3% maximum relative error; i.e., the simulations stopped after 1,000,000 samples were processed.

Power-Delay Product Cost Function Results for $\alpha = 3, \gamma = 1$

Table A.1: Probability of table-miss=.2

	Two controllers with random policy of dispatching			**One controller topology**			
Arrival rate (packets/s)	Average (PDP)	Upper confidence limit	Lower confidence limit	Average (PDP)	Upper confidence limit	Lower confidence limit	compare*
462.96	2.01E+06	2.06E+06	1.97E+06	4.52E+06	4.64E+06	4.40E+06	TRUE
874.487	2.30E+06	2.35E+06	2.24E+06	4.90E+06	4.99E+06	4.80E+06	TRUE
1286.013	2.57E+06	2.62E+06	2.53E+06	5.56E+06	5.71E+06	5.42E+06	TRUE
1697.65	2.93E+06	3.00E+06	2.87E+06	6.33E+06	6.49E+06	6.17E+06	TRUE
2109.067	3.40E+06	3.45E+06	3.33E+06	7.23E+06	7.38E+06	7.09E+06	TRUE
2520.593	4.16E+06	4.28E+06	4.05E+06	8.90E+06	9.19E+06	8.68E+06	TRUE
2932.12	5.30E+06	5.48E+06	5.10E+06	1.09E+07	1.11E+07	1.06E+07	TRUE
3343.647	7.37E+06	7.61E+06	7.12E+06	1.45E+07	1.51E+07	1.40E+07	TRUE
3755.173	1.17E+07	1.26E+07	1.08E+07	2.40E+07	2.60E+07	2.20E+07	TRUE
4166.7	3.92E+07	4.39E+07	3.43E+07	7.38E+07	8.54E+07	6.23E+07	TRUE
	*Lower confidence limit with one controller > Upper confidence limit with random policy?						

Table A.2: Probability of table-miss=.5

	Two controllers with random policy of dispatching			One controller topology			
Arrival rate (packets/s)	Average (PDP)	Upper confidence limit	Lower confidence limit	Average (PDP)	Upper confidence limit	Lower confidence limit	compare*
833.33	4.90E+06	5.01E+06	4.77E+06	9.98E+06	1.02E+07	9.77E+06	TRUE
1574.071	5.30E+06	5.40E+06	5.19E+06	1.11E+07	1.14E+07	1.09E+07	TRUE
2314.812	5.98E+06	6.15E+06	5.80E+06	1.27E+07	1.29E+07	1.24E+07	TRUE
3055.553	6.81E+06	7.00E+06	6.63E+06	1.40E+07	1.43E+07	1.36E+07	TRUE
3796.294	8.12E+06	8.28E+06	7.95E+06	1.65E+07	1.69E+07	1.61E+07	TRUE
4537.036	1.00E+07	1.03E+07	9.80E+06	2.06E+07	2.10E+07	2.02E+07	TRUE
5277.777	1.27E+07	1.29E+07	1.24E+07	2.57E+07	2.63E+07	2.51E+07	TRUE
6018.518	1.78E+07	1.81E+07	1.74E+07	3.64E+07	3.73E+07	3.56E+07	TRUE
6759.259	3.02E+07	3.15E+07	2.87E+07	5.80E+07	6.00E+07	5.61E+07	TRUE
7500	8.41E+07	9.35E+07	7.47E+07	1.61E+08	1.79E+08	1.45E+08	TRUE
	*:Lower confidence limit with one controller > Upper confidence limit with random policy?						

Table A.3: Probability of table-miss=.9

	Two controllers with random policy of dispatching			One controller topology			
Arrival rate (packets/s)	Average (PDP)	Upper confidence limit	Lower confidence limit	Average (PDP)	Upper confidence limit	Lower confidence limit	compare*
462.96	8.50E+06	8.64E+06	8.35E+06	1.77E+07	1.82E+07	1.72E+07	TRUE
874.487	9.47E+06	9.71E+06	9.24E+06	1.99E+07	2.03E+07	1.95E+07	TRUE
1286.013	1.06E+07	1.09E+07	1.04E+07	2.21E+07	2.25E+07	2.18E+07	TRUE
1697.65	1.26E+07	1.29E+07	1.22E+07	2.54E+07	2.60E+07	2.47E+07	TRUE
2109.067	1.48E+07	1.52E+07	1.45E+07	2.99E+07	3.06E+07	2.92E+07	TRUE
2520.593	1.81E+07	1.84E+07	1.77E+07	3.57E+07	3.65E+07	3.49E+07	TRUE
2932.12	2.26E+07	2.31E+07	2.21E+07	4.48E+07	4.59E+07	4.38E+07	TRUE
3343.647	3.22E+07	3.29E+07	3.15E+07	6.27E+07	6.44E+07	6.10E+07	TRUE
3755.173	5.33E+07	5.60E+07	5.08E+07	1.10E+08	1.14E+08	1.06E+08	TRUE
4166.7	1.66E+08	1.86E+08	1.47E+08	3.24E+08	3.65E+08	2.84E+08	TRUE
	*:Lower confidence limit with one controller > Upper confidence limit with random policy?						

Response Time Results with Intermittently Failing Controllers

Table A.4: Probability of table-miss=.2

	One controller with intermittent failures			Two controllers with intermittent failures			
Arrival rate (packets/s)	Average (s)	Upper confidence limit	Lower confidence limit	Average (s)	Upper confidence limit	Lower confidence limit	compare*
2083.3	1.31E-04	1.35E-04	1.28E-04	9.09E-05	9.32E-05	8.87E-05	TRUE
3935.156	1.45E-04	1.52E-04	1.39E-04	8.52E-05	8.66E-05	8.37E-05	TRUE
5787.011	1.61E-04	1.68E-04	1.54E-04	8.90E-05	9.10E-05	8.69E-05	TRUE
7638.867	1.77E-04	1.87E-04	1.67E-04	9.22E-05	9.49E-05	8.95E-05	TRUE
9490.722	2.23E-04	2.38E-04	2.07E-04	9.97E-05	1.03E-04	9.65E-05	TRUE
11342.578	2.43E-04	2.57E-04	2.28E-04	1.09E-04	1.13E-04	1.04E-04	TRUE
13194.433	3.22E-04	3.47E-04	2.97E-04	1.10E-04	1.15E-04	1.05E-04	TRUE
15046.289	4.50E-04	4.97E-04	4.02E-04	1.21E-04	1.26E-04	1.15E-04	TRUE
16898.144	6.98E-04	8.05E-04	5.90E-04	1.37E-04	1.46E-04	1.27E-04	TRUE
18750	2.09E-03	2.48E-03	1.71E-03	1.43E-04	1.50E-04	1.36E-04	TRUE
	*:Lower confidence limit with one controller > Upper confidence limit with two controllers?						

Table A.5: Probability of table-miss=.5

	One controller with intermittent failures			Two controllers with intermittent failures			
Arrival rate (packets/s)	Average (s)	Upper confidence limit	Lower confidence limit	Average (s)	Upper confidence limit	Lower confidence limit	compare*
833.33	3.02E-04	3.10E-04	2.94E-04	2.09E-04	2.14E-04	2.03E-04	TRUE
1574.071	3.37E-04	3.46E-04	3.28E-04	2.04E-04	2.09E-04	2.00E-04	TRUE
2314.812	3.80E-04	3.95E-04	3.66E-04	2.09E-04	2.13E-04	2.05E-04	TRUE
3055.553	4.49E-04	4.67E-04	4.31E-04	2.16E-04	2.22E-04	2.11E-04	TRUE
3796.294	4.98E-04	5.23E-04	4.72E-04	2.23E-04	2.28E-04	2.18E-04	TRUE
4537.036	5.85E-04	6.18E-04	5.52E-04	2.42E-04	2.48E-04	2.36E-04	TRUE
5277.777	7.90E-04	8.41E-04	7.39E-04	2.60E-04	2.68E-04	2.53E-04	TRUE
6018.518	1.09E-03	1.16E-03	1.03E-03	2.80E-04	2.89E-04	2.72E-04	TRUE
6759.259	1.84E-03	2.05E-03	1.64E-03	3.06E-04	3.18E-04	2.93E-04	TRUE
7500	4.41E-03	5.17E-03	3.64E-03	3.35E-04	3.49E-04	3.21E-04	TRUE
	*:Lower confidence limit with one controller > Upper confidence limit with two controllers?						

Table A.6: Probability of table-miss=.9

	One controller with intermittent failures			**Two controllers with intermittent failures**			
Arrival rate (packets/s)	Average (s)	Upper confidence limit	Lower confidence limit	Average (s)	Upper confidence limit	Lower confidence limit	compare*
462.96	5.37E-04	5.52E-04	5.22E-04	3.70E-04	3.81E-04	3.60E-04	TRUE
874.487	6.10E-04	6.22E-04	5.97E-04	3.56E-04	3.64E-04	3.48E-04	TRUE
1286.013	6.90E-04	7.07E-04	6.72E-04	3.57E-04	3.66E-04	3.48E-04	TRUE
1697.65	7.77E-04	7.99E-04	7.55E-04	3.91E-04	4.02E-04	3.79E-04	TRUE
2109.067	9.23E-04	9.57E-04	8.88E-04	4.06E-04	4.17E-04	3.94E-04	TRUE
2520.593	1.14E-03	1.18E-03	1.10E-03	4.29E-04	4.40E-04	4.18E-04	TRUE
2932.12	1.43E-03	1.50E-03	1.37E-03	4.67E-04	4.80E-04	4.54E-04	TRUE
3343.647	1.92E-03	2.03E-03	1.81E-03	5.24E-04	5.39E-04	5.08E-04	TRUE
3755.173	3.05E-03	3.27E-03	2.82E-03	5.48E-04	5.65E-04	5.32E-04	TRUE
4166.7	8.93E-03	1.07E-02	7.18E-03	5.95E-04	6.13E-04	5.77E-04	TRUE
	*:Lower confidence limit with one controller > Upper confidence limit with two controllers?						