

TRAFFIC ENGINEERING IN SOFTWARE-DEFINED  
DATA CENTER NETWORKS FOR IOT

---

*Ayan Mondal*



**TRAFFIC ENGINEERING IN SOFTWARE-DEFINED  
DATA CENTER NETWORKS FOR IOT**

*Thesis submitted to the  
Indian Institute of Technology Kharagpur  
for award of the degree*

*of*

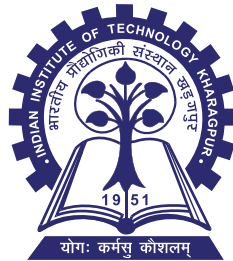
**Doctor of Philosophy**

*by*

**Ayan Mondal**

Under the guidance of

**Professor Sudip Misra**



**Department of Computer Science and Engineering**

**Indian Institute of Technology Kharagpur**

**Kharagpur - 721 302, India**

**December 2020**

© 2020 Ayan Mondal. All rights reserved.



## CERTIFICATE OF APPROVAL

Date: 14/12/2020


Certified that the thesis entitled **Traffic Engineering in Software-Defined Data Center Networks for IoT** submitted by **Ayan Mondal** to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

  
(Member of DSC)

  
(Member of DSC)

  
(Member of DSC)

  
(Supervisor)

  
14.12.2020  
(External Examiner)

  
(Chairman)





भारतीय प्रौद्योगिकी संस्थान खड़गपुर  
खड़गपुर-७२१ ३०२, भारत  
Indian Institute of Technology Kharagpur  
Kharagpur - 721302, India

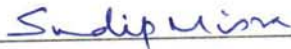
---

Computer Science and Engineering Department

CERTIFICATE

Date: 14/12/2020

This is to certify that the thesis entitled **Traffic Engineering in Software-Defined Data Center Networks for IoT**, submitted by **Ayan Mondal** to Indian Institute of Technology Kharagpur, is a record of *bona fide* research work under my supervision and I consider it worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.



Dr. Sudip Misra

Professor

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Kharagpur - 721 302, India





## DECLARATION

I certify that

- a. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.
- b. The work has not been submitted to any other Institute for any degree or diploma.
- c. I have followed the guidelines provided by the Institute in writing the thesis.
- d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- e. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
- f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

*Ayan Mondal*

---

Ayan Mondal



*Dedicated to  
My Family*



## ACKNOWLEDGMENT

During the period of my PhD research, there are many people whose guidance, support, encouragement and sacrifice have made me indebted for my whole life. I take this opportunity to express my sincere thanks and gratitude to all these people.

First, I would like to express my deepest gratitude to my revered supervisor Professor Sudip Misra for his invaluable guidance and encouragement throughout my work. His constant motivation, support and infectious enthusiasm have guided me towards the successful completion of my work. My interactions with him have been of immense help in defining my research goals and in identifying ways to achieve them. His encouraging words have often pushed me to put in my best possible efforts. Above all, the complete belief that he has entrusted upon me and has instilled a great sense of confidence and purpose in my mind, which, I am sure, will stand me in good stead throughout my career.

It gives me immense pleasure to thank the head of the department Professor Dipanwita Roy Chowdhury for extending me all the possible facilities to carry out the research work. My sincere thanks to my doctoral scrutiny committee chairman Prof. Rajib Mall and all of my doctoral scrutiny committee members Professor Manoj Kumar Tiwari, Professor Sourav Mukhopadhyay, Dr. Arijit Dey, and Dr. Rajiv Ranjan Sahay for their valuable suggestions during my research. I sincerely remember the support of office staffs Swapan Babu, Mithun Da, Bappa Da, Pratap Da, Malay Da, Anup Da, Binod Da, Ajay Da, and Utpal Da.

I would like to express my thanks to Aishwariya, Satendra, Ilora Di, Abhishek Da, Sankar Da, Tamoghna Da, Barun Da, and other SWAN Research Group Members. It is a great fun and source of ideas and energy to have friends like Aishwariya, Satendra, Pradyumna, Samaresh, Anandarup, and Arijit Da for making the stay at Kharagpur ever memorable. I also wish to convey my special thanks to my friends Srinibas and Sridhar for their support and motivation.

Nothing would have been possible without the moral support of family members, who have been the pillars of strength in all my endeavors. Without their unconditional love, sacrifice, support and encouragement, I would have never come this far. I am always deeply indebted to them for all that they have done for me.

Ayan Mondal



## Abstract

In the last two decades, the data generated by different Internet of Things (IoT) applications has increased significantly. The real-time processing, computation, and analysis of these generated data, which is termed as ‘*big-data*’, typically demand support from geographically distributed data centers. These well-connected data centers, forming a data center network (DCN), try to optimize the load distribution among the switches. However, the traditional DCN suffers from unbalanced traffic load and low utilization of network bandwidth, which, in turn, increase energy consumption and degrade the overall performance of the DCN. On the other hand, with the advancement of IoT technologies, different IoT devices are capable of generating and processing a huge amount of data. Hence, there is a need to integrate these IoT devices into the DCN architecture. We argue that this cannot be done using traditional network architecture, as the traditional network devices, such as switches and routers, are not capable of handling different application-specific protocols and heterogeneous IoT devices, due to vendor-specific infrastructure.

We envision that the aforementioned limitations can be resolved by integrating the traditional DCN with the software-defined network (SDN) architecture, which is named as ‘software-defined data center network’ (SD-DCN). SD-DCN assumes the advantages of SDN by decoupling the network control tasks from the tasks of packet forwarding and processing, while dividing them into two parts – the *control plane* and the *data plane*. Due to the presence of heterogeneous IoT applications, SD-DCN needs to handle heterogeneous elephant and mice flows. The existing literature on SDN and DCN focused on the traditional networking issues. However, the implications of the presence of heterogeneous IoT flows need further investigation. In this thesis, we focus on the traffic management strategies in SD-DCN to handle heterogeneous flows generated from the IoT devices and data centers, while ensuring the quality-of-service (QoS) requirements of data traffic in terms of network-delay, -throughput, and -resource utilization. A summary of the major contributions reported in this thesis is presented as follows.

Initially, we focus on designing a probabilistic performance analysis model of an SDN OpenFlow system for data traffic management while considering that the switches are equipped with a finite size buffer. This design helps us to understand the performance of the SDN switches in terms of the probabilities of a packet to be forwarded to the

controller, to the next switch, or to be dropped. Consequently, we designed a scheme to evaluate the optimal buffer size for each switch based on the number of ingress ports and the data traffic pattern, i.e., the packet arrival and processing rates, in an SDN OpenFlow system. Through simulation, we observe that with two times increase in packet processing rate, the packet arrival rate can be increased by 26.15-30.4%. We infer that for an OpenFlow system, the minimum buffer size is 0.75 million packets with the maximum packet arrival and the minimum processing rate of 0.20-0.25 million packets per second (mpps) and 0.30-0.35 mpps, respectively. Thereafter, we consider the presence of heterogeneous applications, which are associated with heterogeneous elephant and mice flows, and have different requirements in terms of network delay and throughput. To increase the flow-wise throughput in SD-DCN, we propose a throughput-optimal data traffic management scheme and evaluate the optimal association of the flows and the available one-hop switches. Additionally, we propose a delay-optimal data traffic management scheme for reducing end-to-end network delay and increasing the overall network throughput in SD-DCN. Simulation result show that the proposed schemes are capable of reducing network delay by 77.8-98.7% while ensuring 24.6-47.8% increase in network throughput compared to the existing schemes. Additionally, the proposed schemes ensure that per-flow delay is reduced by 27.7% with balanced load distribution among the SDN switches.

For the aforementioned schemes, we consider that each flow is associated with a single source-destination pair. However, in SD-DCN, each IoT data traffic can be designated to more than one destination IoT device or data center. Therefore, we focus on designing broadcast and multicast data traffic management schemes in the presence of heterogeneous IoT devices. Moreover, we consider that the IoT devices act as the source nodes and are mobile in nature. In these works, we aim to provide a QoS-guaranteed end-to-end data delivery and maximize the utilization of available network bandwidth, while maximizing the overall network throughput and reducing the flow-specific delay in SD-DCN. Through simulation, we observe that the network throughput increases by 55.32%, while ensuring at least 33% increase in the average bandwidth allocation per IoT device in data broadcast. Additionally, we observe that in data multicast, the network throughput increased by at least 6.13% using the proposed scheme than using the existing schemes, while ensuring at least 21.32% reduction in per-flow delay.

While designing the aforementioned schemes, we considered that a single SDN controller is present in SD-DCN. However, traditional DCN can have multiple network vendors. Therefore, there can be multiple SDN controllers in SD-DCN. In the existing literature, the traditional multi-tenant SDN is visualized to be equipped with an ad-



ditional centralized controller, named ‘proxy controller’ for ensuring optimal flow-table partitioning. However, the presence of the proxy controller gives rise to a single point of failure and bottleneck problems. Hence, to solve these problems, we propose a flow-table partitioning scheme for distributed multi-tenant SD-DCN, while ensuring high throughput and network satisfaction, and reducing flow-setup delay in SD-DCN. Finally, we conclude the thesis, while highlighting the limitations of the aforementioned works and the possible future research directions.

**Keywords:** Traffic Management, Internet of Things, Heterogeneous Flows, Quality-of-service, Game Theory, Software-Defined Network, Data Center Network, Software-Defined Data Center Network



# Contents

Approval	i
Certificate	iii
Declaration	v
Dedication	vii
Acknowledgment	ix
Abstract	xi
Contents	xv
List of Figures	xxi
List of Tables	xxiii
List of Algorithms	xxv
List of Symbols and Abbreviations	xxvii
<b>1 Introduction</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Scope of the Work . . . . .	4
1.3 Contributions . . . . .	7
1.4 Organization of the Thesis . . . . .	9
<b>2 Literature Survey</b>	<b>11</b>
2.1 Performance Analysis of SDNs . . . . .	11
2.2 Resource Management in SDNs . . . . .	13

2.3	Resource Management in DCNs . . . . .	16
2.4	Concluding Remarks . . . . .	18
<b>3</b>	<b>Theoretical Performance Analysis of SDN Switches</b>	<b>21</b>
3.1	Markovian Model: The Justification . . . . .	22
3.2	State Diagram . . . . .	23
3.3	Probabilistic Analysis . . . . .	25
3.3.1	Output Action Probability . . . . .	32
3.3.2	Packet Drop Probability . . . . .	33
3.3.3	Send to Controller Probability . . . . .	34
3.4	Performance Analysis . . . . .	34
3.4.1	Simulation Parameters . . . . .	35
3.4.2	Performance Metrics . . . . .	35
3.4.3	Result and Discussion . . . . .	38
3.5	Concluding Remarks . . . . .	41
<b>4</b>	<b>Buffer Size Analysis of SDN Switches</b>	<b>43</b>
4.1	System Model . . . . .	44
4.1.1	Markovian Process: The Justification . . . . .	45
4.1.2	Packet Flow through an OpenFlow Switch . . . . .	46
4.2	OPUS Scheme: I-M/M/1/K Queue . . . . .	47
4.3	Case Study . . . . .	51
4.3.1	Case I : $I = 1$ . . . . .	51
4.3.2	Case II : $I \geq 2$ . . . . .	52
4.4	Performance Evaluation . . . . .	52
4.4.1	Simulation Parameters . . . . .	53
4.4.2	Performance Metrics . . . . .	54
4.4.3	Results and Discussions . . . . .	55
4.5	Concluding Remarks . . . . .	57
<b>5</b>	<b>QoS-Aware Data Traffic Management</b>	<b>59</b>
5.1	TROD: The Throughput-Optimal Data Traffic Management Scheme . . .	60
5.1.1	System Model . . . . .	60
5.1.2	Justification for Using Evolutionary Game . . . . .	62
5.1.3	Game Formulation . . . . .	63
5.1.3.1	Utility Function of Each Switch . . . . .	64
5.1.3.2	Replicator Dynamics of TROD . . . . .	64

## Contents

---

5.1.3.3	Reduced Sub-Optimal Problem . . . . .	65
5.1.4	Theoretical Analysis: . . . . .	66
5.1.5	Proposed Algorithm: . . . . .	68
5.1.6	Performance Evaluation . . . . .	68
5.1.6.1	Simulation Parameters . . . . .	69
5.1.6.2	Benchmarks . . . . .	69
5.1.6.3	Performance Metrics . . . . .	70
5.1.6.4	Results and Discussions . . . . .	71
5.2	FlowMan: The QoS-Aware Data Traffic Management Scheme . . . . .	72
5.2.1	System Model . . . . .	73
5.2.2	Justification for Using Generalized Nash Bargaining Game . . . . .	75
5.2.3	Game Formulation . . . . .	76
5.2.3.1	Utility Function . . . . .	77
5.2.4	Axioms for Generalized Nash Bargaining Solution . . . . .	79
5.2.5	Existence of Generalized Nash Equilibrium . . . . .	81
5.2.6	Analysis of Generalized Nash Bargaining Solution . . . . .	82
5.2.7	Proposed Algorithm . . . . .	83
5.2.7.1	Complexity Analysis . . . . .	85
5.2.8	Performance Evaluation . . . . .	85
5.2.8.1	Simulation Parameters . . . . .	85
5.2.8.2	Benchmarks . . . . .	86
5.2.8.3	Performance Metrics . . . . .	87
5.2.8.4	Results and Discussions . . . . .	88
5.3	Concluding Remarks . . . . .	89
<b>6</b>	<b>Broadcast Data Traffic Management</b>	<b>91</b>
6.1	System Architecture . . . . .	92
6.2	Proposed D2B Broadcast Scheme . . . . .	94
6.2.1	Justification for Using Single-Leader-Multiple-Follower Stackelberg Game . . . . .	95
6.2.2	Utility Function of Each IoT Device . . . . .	96
6.2.3	Utility Function of Each Switch . . . . .	97
6.2.4	Existence of Equilibrium . . . . .	98
6.2.5	Solution of Proposed D2B . . . . .	99
6.3	Proposed Algorithms for D2B . . . . .	100
6.4	Performance Evaluation . . . . .	102

6.4.1	Simulation Parameters . . . . .	102
6.4.2	Benchmarks . . . . .	102
6.4.3	Performance Metrics . . . . .	104
6.4.4	Results and Discussions . . . . .	106
6.5	Concluding Remarks . . . . .	107
<b>7</b>	<b>Multicast Data Traffic Management</b>	<b>109</b>
7.1	System Model . . . . .	110
7.1.1	Assumptions . . . . .	112
7.2	Dynamic Data Multicasting (D2M) Scheme . . . . .	113
7.2.1	Game Formulation . . . . .	113
7.2.1.1	Justification for Single Leader Multiple Follower Stack- elberg Game . . . . .	114
7.2.1.2	Utility Function of Controller . . . . .	114
7.2.1.3	Utility Function of Each Switch . . . . .	115
7.2.2	Existence of Nash Equilibrium . . . . .	116
7.2.3	Theoretical Analysis of D2M Scheme . . . . .	117
7.3	Proposed Algorithms . . . . .	118
7.4	Performance Evaluation . . . . .	119
7.4.1	Simulation Parameters . . . . .	121
7.4.2	Benchmarks . . . . .	121
7.4.3	Performance Metrics . . . . .	122
7.4.4	Results and Discussions . . . . .	122
7.5	Concluding Remarks . . . . .	123
<b>8</b>	<b>Multi-Tenant Flow-Table Partitioning</b>	<b>125</b>
8.1	System Model . . . . .	126
8.2	BIND: The Proposed Blockchain-Based Flow-Table Partitioning Scheme .	128
8.2.1	Justification for Using Utility Game . . . . .	129
8.2.2	Game formulation . . . . .	129
8.2.3	Replacement Eligibility Factor for Each Flow-Rules . . . . .	130
8.2.4	Utility Function of Each Controller . . . . .	131
8.3	Proposed Algorithms . . . . .	132
8.4	Performance Evaluation . . . . .	136
8.4.1	Simulation Parameters . . . . .	136
8.4.2	Benchmarks . . . . .	136
8.4.3	Performance Metrics . . . . .	137

## Contents

---

8.4.4 Results and discussions . . . . .	138
8.5 Concluding Remarks . . . . .	138
<b>9 Conclusion</b>	<b>141</b>
9.1 Summary of the Thesis . . . . .	141
9.2 Contributions . . . . .	145
9.3 Limitations . . . . .	147
9.4 Future Scope of Work . . . . .	148
<b>Publications</b>	<b>151</b>
<b>References</b>	<b>153</b>





# List of Figures

1.1	Schematic Diagram of SD-DCN Architecture . . . . .	2
1.2	Flowchart of Traffic Management in SD-DCN . . . . .	7
3.1	State Diagram for Packet Flow in an OpenFlow Switch . . . . .	22
3.2	Flowchart for Packet Flow through an OpenFlow Switch . . . . .	24
3.3	Sent to Output Rate of an OpenFlow Switch . . . . .	36
3.4	Packet Drop Rate of an OpenFlow Switch . . . . .	37
3.5	Send to Controller Rate of an OpenFlow Switch . . . . .	38
3.6	Average Queuing Delay of an OpenFlow Switch . . . . .	39
3.7	Average Packet Processing Delay of an OpenFlow Switch . . . . .	40
4.1	OpenFlow Switch with $I$ Ingress Ports/Buffers . . . . .	44
4.2	The Ingress Port/Buffer $i$ of an OpenFlow Switch . . . . .	45
4.3	Maximum Arrival Rate per OpenFlow Switch. . . . .	54
4.4	Maximum Waiting Time per OpenFlow Switch . . . . .	54
4.5	Minimum Buffer Size per OpenFlow Ingress Port . . . . .	55
4.6	Maximum Processing Time per OpenFlow Switch . . . . .	56
5.1	Schematic Diagram of SD-DCN in the Presence of IoT-Devices . . . . .	61
5.2	Volume of Data Traffic Processed by Switches with Varied Number of IoT Devices . . . . .	70
5.3	Throughput of Switches with Varied Number of IoT Devices . . . . .	71
5.4	Population Share of each Switch . . . . .	72
5.5	Issues in Heterogeneous Flow Management in SD-DCN . . . . .	73
5.6	Workflow Diagram of FlowMan . . . . .	83
5.7	Per-Flow Throughput Analysis . . . . .	87
5.8	Network Throughput Analysis . . . . .	87
5.9	Per-Flow Delay Analysis . . . . .	88

5.10	Network Delay Analysis . . . . .	88
6.1	Schematic Diagram for Fat-Tree SD-DCN with IoT Devices . . . . .	92
6.2	Average Bandwidth Allocation per Node . . . . .	104
6.3	Total Bandwidth Utilization . . . . .	104
6.4	Average Delay of the Network . . . . .	105
6.5	Maximum Time Required for Broadcasting 100 Packets . . . . .	105
6.6	Successful Nodes in Broadcasting . . . . .	106
7.1	Schematic Diagram of Fat Tree-based SD-DCN with IoT Devices . . . . .	110
7.2	Performance Analysis of D2M . . . . .	122
8.1	Schematic Diagram of Multi-Tenant SDN . . . . .	126
8.2	Comparison of BIND with Other Schemes . . . . .	137

# List of Tables

3.1	Simulation parameters . . . . .	35
4.1	System Specification . . . . .	53
4.2	Simulation Parameters . . . . .	53
5.1	Simulation Parameters . . . . .	69
5.2	System Specification . . . . .	69
5.3	Simulation Parameters . . . . .	86
6.1	Simulation Parameters . . . . .	103
6.2	Node Capacity Distribution . . . . .	103
7.1	Simulation Parameters . . . . .	119
7.2	Node Capacity Distribution . . . . .	121
8.1	System Specification . . . . .	136
8.2	Simulation Parameters . . . . .	136



# List of Algorithms

5.1	Algorithm for TROD Scheme . . . . .	67
5.2	Data Flow Management in FlowMan . . . . .	84
6.1	IoT Device Registration . . . . .	101
6.2	Optimal Throughput for Each IoT Device $n$ . . . . .	101
6.3	Optimal $p_s(\cdot)$ for Each Switch $s$ . . . . .	102
7.1	Optimal Flow Association Vector . . . . .	119
7.2	Optimal Data-rate for Each Flow . . . . .	120
8.1	FLE: Flow-Rule Election in BIND . . . . .	133
8.2	FRR: Flow-Rule Replacement in BIND . . . . .	134



# List of Symbols and Abbreviations

## List of Symbols

$N$	Number of OpenFlow switches in AMOPE
$B_i$	The $i^{th}$ position of the OpenFlow queue
$F_i$	The $i^{th}$ ingress flow-table
$F_e$	The $e^{th}$ egress flow-table
$p_i$	Probability of having table-hit at the ingress flow-table $F_i$
$p_s$	Probability of packet getting forwarded to OpenFlow switch $s$
$P(F_i)$	Probability of packet entering the ingress flow-table $F_i$
$P(C)$	Probability of the packet getting forwarded to the controller
$P(D)$	Probability of the packet getting dropped
$P(O)$	Probability of the packet being in the output action state
$I$	Number of ingress ports in an OpenFlow switch
$K_i$	Buffer size of ingress port $i$
$\lambda_i$	Mean packet arrival rate at ingress port $i$ in OPUS
$\mu_i$	Mean packet processing rate at ingress port $i$
$q_i$	Probability of a packet getting forwarded to ingress buffer $i$
$q_{m,i}$	Probability of a packet getting forwarded to state $m$ of ingress buffer $i$
$\lambda_{m,i}$	Mean packet arrival rate at state $m$ of ingress port $i$

## List of Symbols and Abbreviations

---

$\mu_{m,i}$	Mean packet processing rate at state $m$ of ingress port $i$
$\mathcal{L}_{s,i}$	Expected number of packets in the system for buffer $i$
$\mathcal{L}_{q,i}$	Expected number of packets in the buffer $i$
$\mathcal{L}_{pr,i}$	Probability that the processing unit of OpenFlow switch is busy
$\mathcal{S}$	Set of available switches
$R_s^{max}$	Maximum flow-rule capacity of switch $s$
$\mathcal{A}$	Set of available IoT devices
$\lambda_i$	The data generation-rate of each data traffic flow $i$ in TROD
$\omega$	Evolutionary iteration
$\sigma$	Evolution controlling factor
$\phi_s(\omega)$	Utility function of switch $s$ in TROD
$y_s(\omega)$	Population share of switch $s$
$G(V, E)$	Flow network graph having $V$ vertices and $E$ edges
$V_I^L$	Set of IoT devices in the layer $L$
$V_S^L$	Set of switches in the layer $L$
$B_{ij}$	Bandwidth capacity of edge $e_{ij} \in E$
$F_n^E(t)$	Set of elephant flows generated by IoT device $n$
$F_n^M(t)$	Set of mice flows generated by IoT device $n$
$F_L(t)$	Number of flows in layer $L$ of the network
$D_k(t)$	Processing delay at switch $k$
$\mathcal{U}_k(\cdot)$	Utility function of each switch $k$ in FlowMan
$\mathcal{A}_s$	Set of IoT devices connected with switch $s$
$T$	Set of time slots in a day
$r_n^{max}$	Maximum data-rate of device $n$
$r_n^{min}$	Minimum data-rate of device $n$
$r_n(t)$	Allocated data-rate of device $n$
$p_s(t)$	Pseudo cost coefficient of switch $s$



## List of Symbols and Abbreviations

---

$\mathcal{M}$	Chunks of data to be broadcasted
$\mathcal{U}_n(\cdot)$	Utility function of IoT device $n$ in D2B
$\mathcal{P}_s(\cdot)$	Utility function of switch $s$ in D2B
$sf_s(t)$	Satisfaction factor of switch $s$ in D2B
$\mathcal{F}_s(t)$	Set of heterogeneous flows associated with switch $s$
$C_s^{rem}$	Remaining capacity of switch $s$ after allocating bandwidth to the data center servers
$\mathcal{D}$	Set of data-servers in SD-DCN
$\rho_s(t)$	Satisfaction factor of each switch $s$ in D2M
$\mathcal{U}_c(\cdot)$	Utility function of controller in D2M
$\mathcal{U}_s(\cdot)$	Utility function of each switch $s$ in D2M
$\mathcal{C}$	Set of SDN controllers
$F_c(\Delta)$	Set of Packet-In messages received by controller $c$ in duration $\Delta$
$\mathbb{P}$	Set of priorities of the flows
$\zeta$	Sustainability of the network
$\mathbb{T}_f$	Tolerable waiting time for each flow-rule $f$
$\mathbb{E}_f$	Replacement-eligibility factor of each flow-rule $f$
$\mathcal{U}_c(\cdot)$	Utility function of controller $c$ in BIND

## List of Abbreviations

SDN	Software-Defined network
DCN	Data Center Network
SD-DCN	Software-Defined Data Center Network
IoT	Internet of Things
mpps	Million packets per second
mp	Million packets

## List of Symbols and Abbreviations

---

AP	Access Point
TCAM	Ternary content-addressable memory
QoS	Quality of Service
kbps	Kilo bytes per second

# Chapter 1

## Introduction

### 1.1 Introduction

In the last two decades, the data generated by different Internet of Things (IoT) applications has increased significantly [1]. The real-time processing, computation [2], and analysis [3] of these generated data, which is termed as ‘*big-data*’, typically demand support from geographically distributed data centers. These well-connected data centers, forming a data center network (DCN), try to optimize the load distribution among the switches. Among different DCN architectures such as fat-tree, BCube, and DCell, we consider the fat-tree DCN architecture for its popularity. Fat-tree DCN follows a hierarchical architecture, where the switches at the aggregation-tier are controlled by the routers at the core-tier. Additionally, the nodes at the edge-tier are controlled by the switches at the aggregation-tier. However, the traditional DCN suffers from unbalanced traffic loads and low utilization of network bandwidth, which, in turn, increase the energy consumption of DCN and degrade the overall performance of DCN. On the other hand, with the advancement of IoT technologies, different IoT devices are capable of generating and processing a huge amount of data. Hence, there is a need to integrate these IoT devices into the DCN architecture. We argue that this cannot be done using traditional

network architecture, as the traditional network devices such as switches and routers, are not capable of handling different application-specific protocols and heterogeneous IoT devices, due to vendor-specific infrastructure.

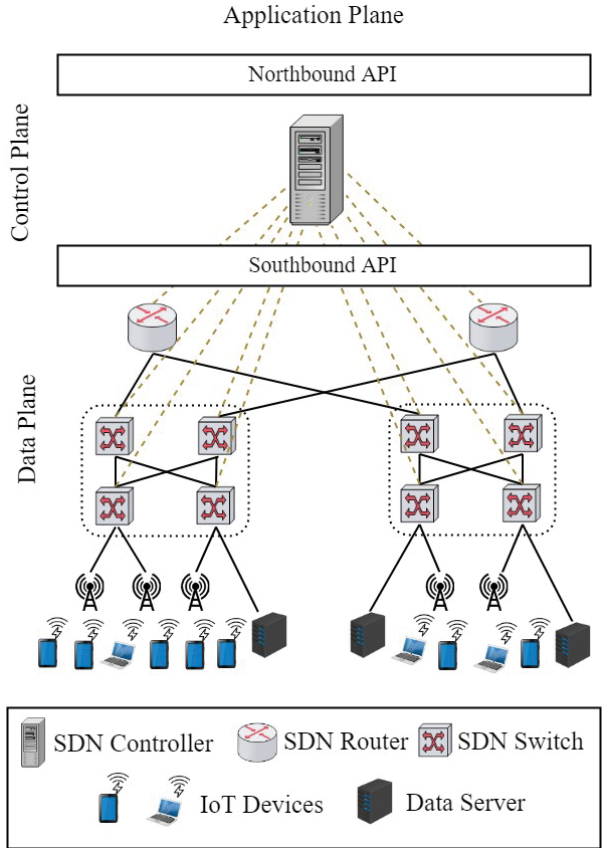


Figure 1.1: Schematic Diagram of SD-DCN Architecture

We envision that the aforementioned limitations can be resolved by integrating the traditional DCN with the software-defined network (SDN) architecture, which is named as ‘software-defined data center network (SD-DCN)’. SD-DCN takes the advantages of SDN by decoupling the network control tasks from the tasks of packet forwarding and processing [4], while dividing into two parts – the *control plane* and the *data plane*, as shown in Figure 1.1. The control plane includes northbound and southbound application programming interfaces (APIs). Presently, OpenFlow is the most popular southbound

## 1.1. Introduction

---

API that enables controller-switch interaction in SDN architecture. An OpenFlow switch contains one or more flow tables to store packet forwarding rules. There are two types of flow tables, namely ingress and egress flow tables. Each flow table contains a set of flow entries. When a packet arrives at the ingress port of an OpenFlow switch, it is matched with the flow entries of the first flow table and with the entries of subsequent flow tables, if required. Matched flow entry provides a set of actions to be executed for the corresponding packet. If no matching entry is found, the packet is dropped or forwarded to the controller for the installation of a new rule depending on the network policy. Each switch in SDN communicates with each external controller via an OpenFlow channel. Additionally, in the new versions of OpenFlow, i.e., OpenFlow version 1.5.0 [5], the presence of hardware and software switches are discussed.

Therefore, analysis of fat-tree DCN in the presence of OpenFlow switch-based SDN is required in order to ensure high quality of services (QoS) of the SD-DCN such as maximum throughput, minimum delay, and proper utilization of available bandwidth, while ensuring minimum energy consumption. Additionally, the network performances in case of data broadcasting and multicasting in SD-DCN in the presence of single and multiple IoT data sources also need to be assessed in fat-tree SD-DCN. On the other hand, there is a requirement to revisit the rule placement, controller and OpenFlow switch placement in the context of SD-DCN. Hence, different schemes need to be designed in order to address the aforementioned scenarios.

The rest of this Chapter is organized as follows. In Section 1.2, the scope of this Thesis, while focusing on traffic engineering in SD-DCN, is presented. The objectives of the work are also presented in this section. Thereafter, Section 1.3 presents the specific contributions of this Thesis. Finally, we conclude this Chapter while mentioning the organization of the Thesis in Section 1.4.

## 1.2 Scope of the Work

We identify the necessity of designing different traffic engineering schemes for SD-DCN in the presence of heterogeneous – elephant and mice – flows, in terms of analyzing the system performance and ensuring network QoS, viz., high throughput and low delay. In particular, we focus on theoretical performance analysis and data traffic management to satisfy the QoS requirements of the heterogeneous flows in SD-DCN for IoT.

**Theoretical Analysis of SDN System:** In the existing literature, researchers proposed different schemes and architectures for SDN, viz., [4,6–8], while considering OpenFlow protocol and OpenFlow switches. The proposed approaches require a substantial amount of execution time depending on the available hardware. This necessitates the development of an analytical model to evaluate the performance of SDN architecture before the actual implementation or simulation. Although, in the existing literature, there are a few works that provide analytical models for performance analysis of an OpenFlow-enabled network [9,10], none of these models define the probabilistic bounds of the OpenFlow protocol version 1.5.0 [5]. Moreover, the proposed models do not evaluate significant network performance parameters such as throughput, average packet processing time, average delay, and packet drop count. However, based on the existing literature, we argue that the performance of the schemes, which are proposed for SDN, highly depends on the optimum values of buffer size, packet arrival, and processing rates. To the best of our knowledge, in the existing literature, there is no analytical model for the evaluation of the minimum buffer size of an SDN switch. Additionally, there is a need for evaluating the relations among the maximum arrival rate, the minimum processing rate, and the minimum buffer size and to estimate the maximum packet waiting time in an OpenFlow switch. These necessitate the design of an analytical model to evaluate the minimum buffer size requirement of an OpenFlow switch for ensuring QoS with minimum packet drop.

## 1.2. Scope of the Work

---

**Data Traffic Management:** Since SD-DCN is envisioned to handle heterogeneous data traffic from IoT devices and data centres, efficient data traffic management in SD-DCNs is a significant as well as challenging task. In the existing literature, researchers focused on data traffic management in the context of DCN, viz. [11–14], and SDN, viz. [6,15,16]. However, these existing schemes are not capable of ensuring balanced data traffic in the presence of heterogeneous data flows. This is because, in the presence of heterogeneous IoT flows, the nature of flows also need to be considered while distributing the flows to ensure network QoS in terms of throughput and delay. Hence, we argue that the existing schemes are not suitable to be used in the context of SD-DCNs. This necessitates the design of efficient data traffic management schemes for SD-DCNs, in order to ensure high network throughput and low network delay. Additionally, we aim to ensure the optimal usage of network bandwidth and balanced data traffic in SD-DCN.

Furthermore, in the existing literature, most researchers considered the traditional fat-tree DCN architecture, which ensures multiple equal-cost paths between any pair of hosts located in different sub-trees [17] of DCN. Therefore, the traditional fat-tree DCN architecture ensures high bandwidth inter-connectivity and path multiplicity. A significant problem in traditional DCN is the unbalanced traffic distribution. Uneven traffic load causes inefficient data parallelization, inferior network performance, and degradation of performance quality of the nodes [18]. Thus, researchers proposed different scheduling techniques for data unicasting [19,20] and multicasting [21] in traditional DCN. However, none of the works addressed the issues of efficient broadcasting in DCN. Hence, there is a need for a load balancing scheme for broadcasting in DCN, since, in the presence of multiple IoT devices [22], broadcasting big-data in real-time is a challenge. We refer to real-time data broadcasting in order to signify that a large amount of data, generated by the set of IoT devices, is being broadcasted to each node at the edge-tier of the fat-tree DCN. Hence, we plan to design an optimal bandwidth distribution and data parallelization scheme for SD-DCN to improve the network performance, while con-

sidering the presence of IoT devices. We consider that, in SD-DCN, the separation of the control plane from the switches and the routers helps to improve the network performance, while maximizing the throughput of the network and minimizing the overall delay. Furthermore, in the presence of mobile IoT devices at the edge-tier, the problem of big-data multicasting in the fat-tree DCN also needs to be revisited in the context of SD-DCN, while ensuring the QoS of the network. In this case, we consider that the data generated from the mobile IoT sources at the edge-tier needs to be multicasted to the subset of available IoT devices at the edge-tier and the data centers. Hence, there is a need for designing an optimal data multicasting scheme for fat-tree SD-DCN.

**Multi-Tenant Flow-Table Partitioning:** As SD-DCN follows a geographically distributed architecture, we consider the presence of multiple controllers, which reduces the probability of single-point failure and ensures scalability. We studied that, in the existing literature, a few works focused on flow-table partitioning in multi-tenant SDN [23–25]. However, these schemes overlooked the presence of heterogeneous flows, hence, these are not suitable for multi-tenant SD-DCN. Therefore, throughput-optimal flow-table partitioning in multi-tenant SD-DCN in the presence of heterogeneous IoT applications, while minimizing flow-rule replacements and maximizing network sustainability, is a pressing issue which needs to be addressed.

The flowchart of the work is presented in Figure 1.2. To summarize, the *objectives* of the work are as follows:

1. Design of a probabilistic packet-centric performance analysis of OpenFlow switch.
2. Design of a buffer size evaluation scheme for ensuring QoS in SD-DCN.
3. Design of a throughput-optimal dynamic data traffic management scheme for SD-DCN in the presence of heterogeneous flows.
4. Design of a delay-aware dynamic data traffic management scheme for SD-DCN in



### 1.3. Contributions

---

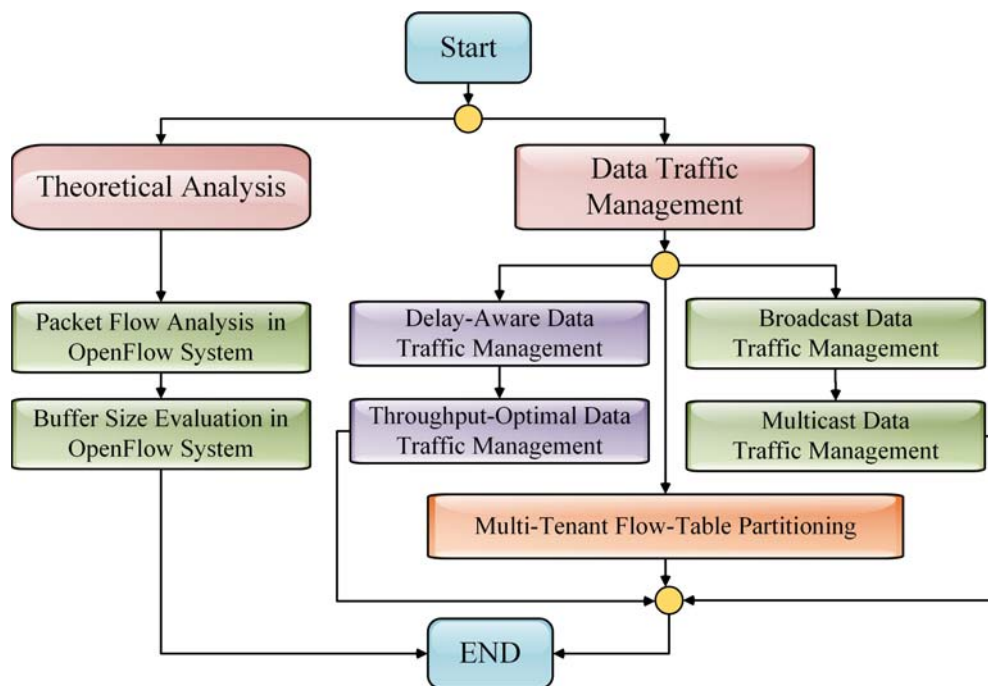


Figure 1.2: Flowchart of Traffic Management in SD-DCN

the presence of heterogeneous flows.

5. Design of a dynamic data broadcasting scheme in the presence of mobile IoT sources in fat-tree SD-DCN.
6. Design of a dynamic data multicasting scheme in the presence of mobile IoT sources in fat-tree SD-DCN.
7. Design of a flow-table partitioning scheme in distributed multi-tenant SD-DCN.

### 1.3 Contributions

The major contribution of this work are listed as follows:

1. We propose a queueing theory-based Markovian model, named AMOPE, to emulate the behavior of an OpenFlow system based on OpenFlow switch specification

version 1.5.0 and evaluate the probabilistic bounds of different performance metrics such as probabilities of packets being dropped, and packets getting forwarded to the output port and to the controller, when an incoming flow of packets passes through the switch.

2. We propose an analytical scheme, named OPUS, for buffer bound evaluation of an OpenFlow system. Additionally, we propose a queueing model based on  $C-M/M/1/K/\infty$  queue for an OpenFlow system following the OpenFlow specification version 1.5.0. Further, we calculate the minimum buffer size requirement of an OpenFlow switch, theoretically.
3. We design a game theory-based dynamic data traffic management scheme for minimizing network delay and maximizing network throughput in SD-DCN in the presence of heterogeneous IoT devices. We use an evolutionary game-theoretic approach to decide the optimal data traffic volume which needs to be handled by the switches, while considering that the data generation rate for each IoT device is known *a priori*.
4. We introduce a QoS-aware stochastic data flow management scheme for SD-DCN in the presence of heterogeneous flows. We use a generalized Nash bargaining game to decide the Pareto optimal data rate to be allocated to each SDN switch, while considering the heterogeneous flows within the one-hop network.
5. We propose a single-leader-multiple-followers Stackelberg game-based scheme, as a pseudo-Cournot competition, for broadcasting big-data in fat-tree SD-DCNs with mobile IoT devices. We divide the entire network into multiple blocks. In each block, an individual switch acts as the leader, and the devices, which are connected to the switch, act as followers. Each switch distributes the available capacity among the connected IoT devices in order to achieve high performance with optimal throughput and delay for big-data broadcasting in fat-tree SD-DCN.

#### 1.4. Organization of the Thesis

---

6. We formulate a single-leader-multiple-followers Stackelberg game-based multicast traffic management scheme to ensure high utilization of network capacity and efficient load balancing in SD-DCN. We consider that the controller acts as the leader and installs the flow-rules in the SDN switches. The controller also decides the source node of the flow for each destination. On the other hand, the SDN switches, which act as the followers, decide their respective strategies, non-cooperatively. The followers help the controller to manage the network properly by deciding the amount of bandwidth to be allocated for each flow and optimizing the usage of overall capacity.
7. We propose a blockchain-based flow-table partitioning scheme for distributed multi-tenant SDN. Using blockchain, we ensure that the controllers are synchronized and cooperative in nature. We use utility game to propose a distributed algorithm for flow-rule election, where each controller distributively identifies the flow-rules' replacement eligibility factors, and elects a single flow-rule for replacement. Thereafter, we consider another utility game-based centralized algorithm for flow-rule replacement to be performed by the controller receiving the Packet-In message.

#### 1.4 Organization of the Thesis

The rest of the Thesis is organized as follows:

- **Chapter 2 – Literature Survey:** The related works on performance analysis in SDN and resource management schemes in SDN and DCN are surveyed in this Chapter.
- **Chapter 3 – Theoretical Performance Analysis of SDN Switches:** In this Chapter, the performance of packet flow through an OpenFlow switch in SD-DCN is analyzed to define the probabilistic bounds of the performance metrics of the OpenFlow switch.

- **Chapter 4 – Buffer Size Analysis of SDN Switches:** In this Chapter, the optimum buffer size of an OpenFlow switch in order to ensure QoS in OpenFlow systems is presented.
- **Chapter 5 – QoS-Aware Data Traffic Management:** The optimal data traffic management schemes are presented in this Chapter, while considering different QoS parameters such as network-throughput and delay.
- **Chapter 6 – Broadcast Data Traffic Management:** This Chapter presents data traffic management schemes for IoT data broadcasting in fat-tree SD-DCN.
- **Chapter 7 – Multicast Data Traffic Management:** This Chapter presents data traffic management schemes for multicasting in fat-tree SD-DCN in the presence of mobile IoT devices.
- **Chapter 8 – Multi-Tenant Flow-Table Partitioning:** In this Chapter, a blockchain-based flow-table partitioning scheme, named BIND, for distributed multi-tenant SD-DCN is presented.
- **Chapter 9 – Conclusion:** This Chapter contains the summary of the Thesis while citing a few research directions.

## Chapter 2

# Literature Survey

In this Chapter, we survey the related literature on traffic engineering schemes for DCNs and SDNs in detail. The existing literature related to traffic engineering of SD-DCNs is divided into two categories — theoretical analysis of SDNs, and resource management in SDNs and DCNs.

The rest of the chapter is organized as follows. Section 2.1 presents the related performance analysis schemes proposed for SDNs in the existing literature, while Sections 2.2 and 2.3 discuss the existing literature on resource management schemes in SDNs and DCNs, respectively. Finally, Section 2.4 concludes the chapter.

### 2.1 Performance Analysis of SDNs

Although, in existing literature, the researchers explored different aspects in the context of OpenFlow and SDN, few works addressed the performance analysis of an OpenFlow-enabled SDN architecture. Jarchel *et al.* [10] modeled the OpenFlow architecture as an M/M/1 forward queueing system and an M/M/1-S feedback queueing system. This model measures the delay at an OpenFlow switch and estimates the total sojourn time of a packet and the probability of dropped packets. Additionally, this model studies the probability of a packet getting blocked by a heavily loaded controller. This work is

based on OpenFlow version 1.0.0, where each switch has a single flow table. The authors considered TCP traffic instead of UDP. Also, they considered that only the first packet header of a new flow is sent to the controller. The authors assumed that the queue length of a switch is infinite. However, according to the OpenFlow version 1.5.0 [5], each switch has multiple flow tables (both ingress and egress) with more number of match fields. In another work, Azodolmolky *et al.* [9] modeled SDN based on network calculus. This model analyses network performance from an SDN controller's perspective and depicts controller-switch interactions. In another work, Metter *et al.* [16] formulated an M/M/ $\infty$  queuing system-based analytical model for analyzing a trade-off between signaling rate and switch table occupancy and calculated an optimum flow-rule time-out period.

Eager and Sevcik [26] studied the performance bounds for single-class queuing networks with fixed rates and delay service centers using mean-value analysis. The authors claimed that the performance bounds ensure the accuracy of the model. There is a need for similar probabilistic analytical models to evaluate the performance bounds for OpenFlow switch in SDN. On the other hand, Garrido *et al.* [27] focused on a Markov chain-based semi-analytical model for evaluating the performance of sparse network coding. The authors used an absorbing Markov process for the work. Bergstrom *et al.* [28] proposed a Markov chain-based analytical model for an optical shared-memory packet switch. The authors evaluated the throughput and probability of packet loss of the system. Similarly, for an OpenFlow switch, there is a need for packet-centric analysis and for evaluating the probabilistic bounds of the performance metrics. Additionally, Bianco *et al.* [29] compared the performance of OpenFlow switching with that of link-layer Ethernet switching and network layer IP routing. The authors used the packet latency and the forwarding throughput as major performance indicators.

On the other hand, Rich and Schwartz [30] studied the buffer limitation in computer-communication networks, while exploring the advantages of buffer sharing among the communication nodes. Kekre and Saxena [31] proposed an analytical queuing model

## 2.2. Resource Management in SDNs

---

for an integrated digital voice-data system with synchronous time-division multiplexing. The authors analyzed different performance metrics such as overflow probabilities, buffer size, and expected queueing delay due to buffering. Luan [32] analyzed buffer behavior in DCNs. Average buffer size in Intermittently Connected Networks (ICNs) was estimated by Cello *et al.* [33]. In another paper, Manoj *et al.* [34] studied the performance of a buffer-aided multi-hop relaying system using a Markov chain. The authors evaluated analytical expressions for the steady-state probability vector for a three-hop buffer-aided system. Asheralieva and Miyanaga [35] presented an analytical study of optimum buffer size requirement, while estimating the buffer status information for a Long Term Evolution-Advanced (LTE-A) network. Jagannathan *et al.* [36] estimated the queue-length of a system with a single server and two parallel queues — heavy-tailed and light-tailed traffic.

## 2.2 Resource Management in SDNs

Concerning the improvement of OpenFlow-enabled networks, in the existing literature, Reitblatt *et al.* [37] addressed the issues of consistent network updates. The authors proposed a set of abstract operations to change the network configuration, such that, each incoming packet follows either the old configuration or the new one. Bera *et al.* [38] studied different aspects of resource allocation in SDN for IoT. Additionally, Katta *et al.* [7] provided a consistent network update mechanism that addresses the trade-off between rule-space overhead and update-time. Congdon *et al.* [6] proposed a per-port optimization technique to reduce switch latency and power consumption. Saha *et al.* [39] proposed a flow-rule aggregation scheme for SDN, while focusing on the problem of over-subscription. The authors used a key-based aggregation policy to reduce the number of flow rules. In another work, Maity *et al.* [40] proposed a tensor-based flow-rule aggregation scheme in SDN. Meiners *et al.* [8] proposed a technique to compress flow-table entries and increase storage space in an OpenFlow switch. Huang *et al.*

[41] proposed a rule multiplexing scheme to reduce the usage of TCAM memory while maintaining QoS constraints. The authors formulated a joint optimization problem while considering route engineering and rule placement. In another work, Sadeh *et al.* [42] proposed a scheme, named Bit Matcher, to reduce the TCAM memory usage for a given set of flow-Traffic-aware rules.

Rottenstreich *et al.* [43] proposed a traffic splitting scheme for switches while considering the heterogeneity of the network paths or servers and the limited capacity of the flow-tables. Wang *et al.* [44] proposed an SDN-based network storage while having no physical storage. Li *et al.* [45] proposed to store the packet header instead of the entire packet in the buffer of an SDN switch for reducing the communication overheads of SDN. Hayes *et al.* [46] studied the traffic-classification in SDN. In another work, Mogul *et al.* [15] used hashing to reduce flow table lookups in an OpenFlow switch. Saha *et al.* [47] proposed a QoS-aware routing scheme for SDN, while maximizing end-to-end delay. The authors considered different types flows in terms of delay- and loss-sensitivity. Bera *et al.* [48] proposed an SDN-based wireless sensor network for provisioning application-aware service in IoT. Additionally, Misra *et al.* [49] designed a protocol selection scheme for SDN-based wireless sensor networks. In this work, initially, the authors selected the most appropriate protocols for the situations and executed the actual deployment in the next phase. In another work, Bera *et al.* [50, 51] studied a mobility-aware SDN and attempted to maximize the overall network performance.

Agarwal *et al.* [52] studied traffic handling in SDN. The authors showed that having a centralized view of the network, the controller is capable of reducing the delay and packet loss in data traffic. Tseng *et al.* [53] studied the problem of path stability in hybrid SDN. The authors calculated the routes locally to reduce computational complexity, thereafter, used a centralized scheme to re-evaluate the routes for gaining stability. Misra and Bera [54] proposed a task offloading scheme for SDN-based fog network. The authors minimized the delay in task offloading and computation, while selecting the op-



## 2.2. Resource Management in SDNs

---

timal number of fog nodes. Misra and Saha [55] extended the aforementioned work of task offloading in SDN-based fog networks, while considering multi-hop network flows. Moradi *et al.* [56] proposed an efficient traffic engineering scheme, named DRAGON, for SDN-based ISP networks having different types of network links and switches. In DRAGON, the flow optimization problem is broken down into sub-tasks having different objectives and the sub-tasks are executed in parallel to reduce complexity. In another work, Allybokus *et al.* [57] proposed a fair resource allocation scheme in the presence of multiple network paths in a distributed SDN scenario, using the alternating direction method of multipliers. Sanvito *et al.* [58] proposed a scheme for deciding the time frame to reconfigure flow-tables, while considering overlapping data flow paths.

Mondal *et al.* [59] proposed a scheme to ensure high throughput in SDN while assuming that the volume of data to be generated is known *a priori*. The authors optimally distributed the traffic load among the switches and ensured high network throughput. Tahaei *et al.* [60] presented an SDN-based flow management scheme for data center networks with multiple controllers and selected an optimal number of switches. In another work, Görkemli *et al.* [61] proposed a novel distributed dynamic control plane architecture in which the switches communicate with their controllers through a virtual overlay network. The authors also proposed the introduction of a “control flow table” to manage the dynamic control plane traffic. Another dynamic traffic engineering scheme was proposed by Bera *et al.* [62] in which the authors attempted to reduce the control overheads by reducing the number of messages sent to the controller. The authors proposed a greedy heuristics-based approach to determine the optimal number of candidate switches (with higher TCAM) necessary to reduce the involvement of the controller.

Here, we discuss the existing schemes focusing on flow-rule processing in SDN. Mao *et al.* [63] proposed a Convolutional Neural Networks (CNNs)-based intelligent traffic management scheme. The authors considered that the controller has high computational resources. Rottenstreich *et al.* [64] studied the shared multi-core processing scheme and

evaluated a trade-off between the number of allocated cores and the associated delay in the context of network virtualization. In another work, Rottenstreich and Tapolcai [65] proposed a limited-size classifier set to accommodate the flow-rules in limited TCAM memory. Singh *et al.* [66] proposed a hash-based flow-table to reduce the flow-table lookups. In another work, Aujla *et al.* [67] proposed a traffic flow management scheme in SDN.

Furthermore, Bera *et al.* [68] studied the problem of assigning controllers for each flow in SDN using a dynamic stable-matching game. Blenk *et al.* [23] presented a survey of multi-tenant SDN and divided the existing literature into two categories, such as, hard and soft partitioning of flow-tables. In another work, Caria *et al.* [24] proposed an SDN topology partitioning scheme by introducing SDN border nodes, which are responsible for ensuring connectivity among two controllers. The authors considered that each controller has access to a mutually exclusive set of SDN switches. Lin *et al.* [25] proposed a scheme for multi-tenant SDN while considering the presence of a proxy controller.

### 2.3 Resource Management in DCNs

In existing literature, researchers studied *Fat-tree DCNs* [18, 69]. Fat-tree DCN follows a hierarchical architecture. The switches at the aggregation-tier are controlled by the routers at the core-tier. Additionally, the nodes at the edge-tier are controlled by the switches at the aggregation-tier. We divide the existing literature on the data transmission in DCN into two categories — (1) broadcasting and (2) unicasting and multicasting.

Chen *et al.* [11] surveyed the challenges in generation, acquisition, storage, and processing of data. They also mentioned various applications involving big-data such as – enterprise management, IoT, and social networks, while considering different medical applications and smart grid. Muntean *et al.* [12] proposed a quality-oriented adaptation scheme for ensuring delivery of high bit-rate multimedia streams to the users using the

### 2.3. Resource Management in DCNs

---

IP network efficiently. In another review article, Jagadish *et al.* [1] mentioned different challenges for understanding a huge amount of data while citing a case study about cleaning, analyzing, and interpretation of data or information. Wu *et al.* [13] studied a big-data broadcasting scheme for distributed systems. The authors considered that the source node has the maximum bandwidth or capacity, and modeled the network as a lock-step broadcast tree (LBST). Yu *et al.* [14] surveyed different networking aspects of big-data, such as, distributed and heterogeneous networks. The authors also studied different schemes on big-data representation. On the other hand, Liu *et al.* [70] proposed a neighbor-based probabilistic broadcast scheme for data distribution among the mobile IoT devices. The authors determined the re-broadcast probability, while considering the neighborhood nodes and the adaptive connectivity factor.

Lau *et al.* [71] proposed an Audience-Driven Live TV Scheduling (ADTVS) framework using 4G LTE broadcast in order to improve the traditional live television broadcasting system. Zarb and Debono [72] proposed a scalable free-viewpoint television broadcast architecture for long-term evolution cellular networks. Lakhlef *et al.* [73] proposed agent-based broadcast protocols for mobile IoT devices, while considering parallel data broadcasting with a limited channels. Based on the availability of communication channels, the network is partitioned into several groups, where each group has a group-leader, i.e., agent. Ahlgren *et al.* [74] surveyed data transfer in the context of Information-Centric Networking (ICN). Unlike DCN, in ICN, the data files are accessed by the user by their name or identifier, instead of the name of the host device. On the other hand, in DCN, the user accesses the data file by the host identifier. Hence, we argue that the schemes designed for ICN are not applicable to data broadcasting in DCN. Trestian *et al.* [75] studied a network selection scheme, named E-PoFANS, for multimedia delivery in ad-hoc networks. Paul *et al.* [76] studied the optimal server provisioning problem in DCN and proposed two different schemes — for minimizing operational cost and for minimizing capital and operational cost, jointly, based on a discrete-time model.

A few research works exist on data unicasting and multicasting in fat-tree based DCNs. Guo and Yang [21] studied multicasting in DCNs with fat-tree topology. The authors claimed that their work is one of the pioneering work which explores multicasting in fat-tree based DCNs. Iyer *et al.* [77] proposed a multicast routing scheme by reducing the routing tree for a group. Raiciu *et al.* [78] proposed a multipath transmission control protocol (MPTCP) in DCN for data unicasting. They showed that, using MPTCP, the workload is balanced properly in fat-tree topology-based DCNs. Zhu *et al.* [79] designed a multicasting scheme for DCN, while considering the packet processing and flow replication as bottlenecks. Chiu and Lau [80] proposed a scheme for efficient multicast and broadcast services using transmitter-side channel state information. In another work, Al-Fares *et al.* [81] studied a dynamic flow scheduling (HEDERA) scheme for data multicasting, while aggregating network resources. They claimed that HEDERA performs 113% better than static load balancing in DCNs. Curtis *et al.* [82] also studied multicasting traffic pattern in DCNs.

### 2.4 Concluding Remarks

In this Chapter, we reviewed several relevant research works representing some of the emerging trends in theoretical analysis and resource allocation in SDN and DCN, where data traffic management plays an important role. For instance, traditional DCN tries to optimize the load distribution among the switches. However, it suffers from unbalanced traffic load and low utilization of network bandwidth, which, in turn, increases energy consumption and degrades the overall performance of the DCN. Additionally, the traditional network devices, such as switches and routers, are not capable of handling different application-specific protocols and heterogeneous IoT devices, due to vendor-specific infrastructure. On the other hand, the existing works on SDN focused on routing and the efficient utilization of TCAM, however, none of these works considered the presence of heterogeneous flows or application in SDN. Similarly, none of the works in DCN con-

## 2.4. Concluding Remarks

---

sidered the heterogeneity in applications. Hence, in the context of SD-DCN, the data traffic management schemes need to be revisited, while considering the presence of heterogeneous applications or flows. Moreover, the analytical models proposed for SDN in the existing literature considered the complexity of the flow management. However, the performance analysis and evaluation of different physical parameters, such as data flow rates and buffer size, are not explored in the existing literature, which necessitates the design of theoretical models to address the aforementioned problems. Furthermore, the handling of data traffic in distributed multi-tenant SD-DCN is also not explored in the existing literature.



## Chapter 3

# Theoretical Performance Analysis of SDN Switches

In this Chapter, we present a Markovian model-based analytical scheme, named AMOPE, to analyze the behavior of an OpenFlow switch based on its specification version 1.5.0 [5]. In AMOPE, we consider that each controller is connected to multiple switches in SD-DCNs. For each switch, we estimate the performance of the switch using the metrics such as packet drop and throughput, while considering packet queuing, ingress and egress processing steps. Considering that the OpenFlow switch follows packet-level services, in AMOPE, the switch takes each packet as an individual entity, despite taking flow-specific data. We present a Markovian analysis of packet flow through an OpenFlow switch using Markov chain [83,84]. According to the OpenFlow switch specification version 1.5.0, we consider that such a switch has three parts — (a) the switch queue, (b) ingress processing unit, (c) and egress processing unit. Additionally, to design AMOPE, we assume that — (1) each mouse flow comprises a few number of packets; (2) the packet arrival process follows a Poisson distribution; and (3) the packet inter-arrival time follows an exponential distribution.

This Chapter is organized as follows. We describe the state diagram and the prob-

abilistic analysis of each part of an OpenFlow switch in Sections 3.2 and 3.3, while providing a justification for using Markov model in Section 3.1. Section 3.4 discusses the performance evaluation of packet-centric analysis of OpenFlow switch using AMOPE. Finally, Section 3.5 concludes this Chapter.

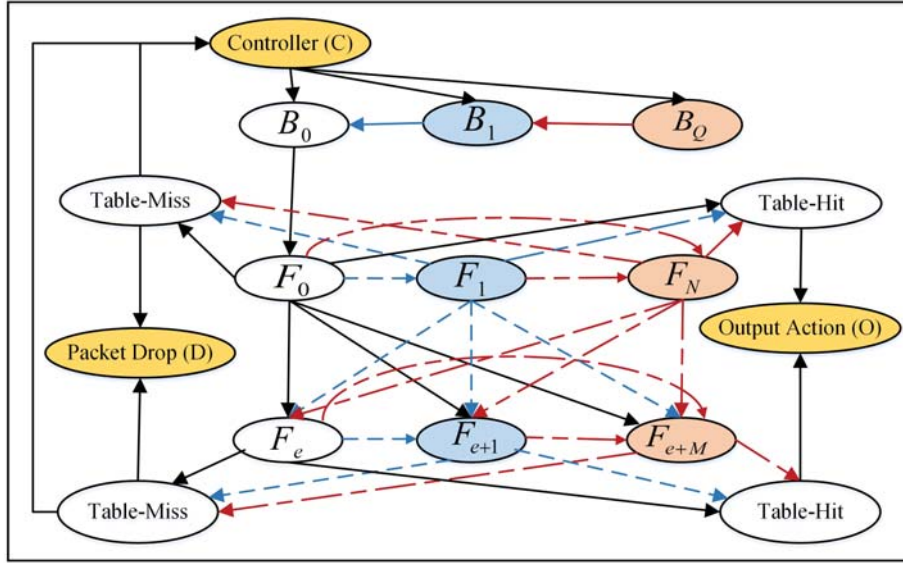


Figure 3.1: State Diagram for Packet Flow in an OpenFlow Switch

### 3.1 Markovian Model: The Justification

We studied the behavior of an OpenFlow switch using *Markovian chain* [85, 86], as it follows the following Markov properties:

- 1) Each packet is processed individually, i.e., the behavior of an OpenFlow switch is *memoryless*.
- 2) Packet processing in an OpenFlow switch is a stochastic process having Markov property  $P(x_{n+1}|X_n, X_{n-1}, X_{n-2}, \dots, X_1) = P(x_{n+1}|X_n)$ , where  $X_n, X_{n-1}$ , and  $X_{n+1}$  are the present, immediate past, and future state of a Markovian process, respectively.



## 3.2 State Diagram

We consider that when a packet is sent from the controller to the OpenFlow switch, the packet gets queued, initially, before entering the ingress processing unit of an OpenFlow switch, as shown in Figure 3.2. The switch has a queue of length of size  $(Q + 1)$ , and each  $i^{th}$  position of the OpenFlow queue is denoted as a separate state  $B_i$ , where  $0 \leq i \leq Q$ , as shown in Figure 3.1. If the packet gets queued at position  $i$ , it waits for a finite duration of time in order to reach the  $0^{th}$  position of the queue,  $B_0$ . Thereafter, the packet enters the ingress processing unit of the switch and searches for a match at the  $0^{th}$  ingress flow-table,  $F_0$ . Hence, a *table-hit* may be possible, signifying a match being found in the table, or a *table-miss*, where no match is found. In the case of table-hit, the OpenFlow switch executes one of the following instructions:

1. The packet goes to another ingress table  $F_i$ , where  $(i > 0)$ .
2. The action mentioned in the action field of the flow entry gets executed.

On the other hand, in the case of table-miss, the packet follows one of the following possibilities:

1. Pass to another ingress flow-table  $F_i$ , where  $(i > 0)$ , according to the table-miss flow entry.
2. Pass to the controller, according to the table-miss flow entry.
3. Drop the packet according to the table-miss flow entry.
4. Drop the packet if there is no table-miss flow entry.

After reaching the ingress flow-table  $F_i$ , the packet is matched against the flow-table entries. If there is table-hit, either the packet gets forwarded to another ingress flow-table  $F_j$ , where  $j > i$ , or instructions are executed according to the flow-table entry, as

### 3. Theoretical Performance Analysis of SDN Switches

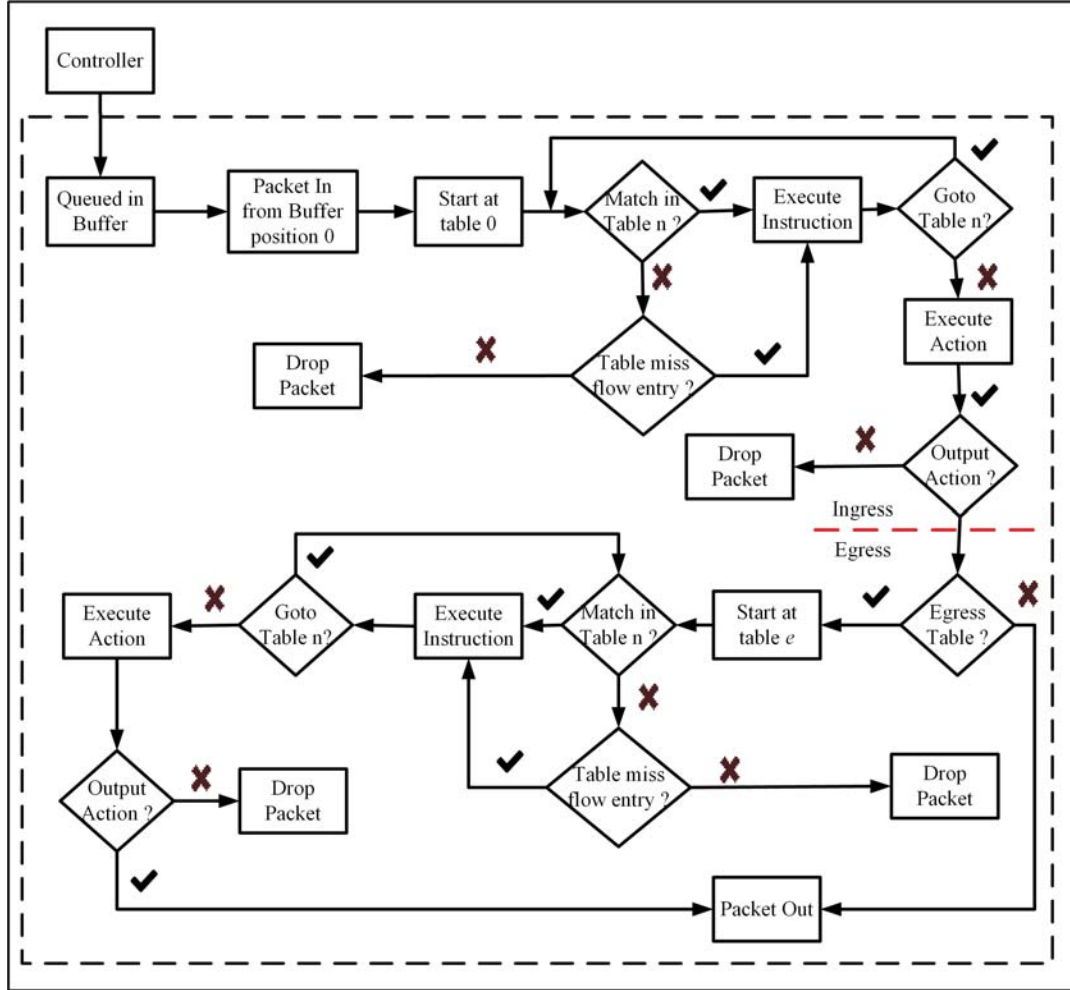


Figure 3.2: Flowchart for Packet Flow through an OpenFlow Switch

discussed earlier. On the other hand, in case of table-miss, the packet gets forwarded either to another ingress flow-table  $F_j$ , where  $j > i$ , or to the controller, according to the table-miss flow entry, or gets dropped, as mentioned earlier.

After the processing of the packet at the ingress processing unit, and the output action is taken according to the table-hit at the ingress flow-table, if the *egress flag* is set for that packet, the packet enters the *egress processing unit*, as shown in Figure 3.2. Additionally, as shown in Figure 3.1, a packet can enter the egress processing unit from

### 3.3. Probabilistic Analysis

---

any of the available ingress flow-tables. Once the packet enters the egress processing unit, the packet gets forwarded to the egress flow-table  $F_e$ , as shown in Figures 3.2 and 3.1. Hence, the packet is matched against the flow-table entries, as mentioned for the ingress processing unit. For egress table-hit at  $(e + i)^{th}$  egress flow-table,  $F_{e+i}$ , where  $0 \leq i < m$ , the packet either gets forwarded to another egress flow-table  $F_{e+j}$ , where  $i < j \leq m$ , or forwarded for executing action set as mentioned in the action field of the matched entry. On the other hand, in the case of table-miss, one of the following options is executed:

1. The packet gets forwarded to the next egress flow-table  $F_{e+j}$ , from the egress flow-table  $F_{e+i}$ , where  $i < j \leq m$ , according to the egress table-miss flow entry.
2. The packet is sent to the controller, according to the egress table-miss flow entry.
3. The packet is dropped, according to the egress table-miss flow entry.
4. The packet is dropped if there is no egress table-miss entry.

If the packet is sent to the controller, the controller handles the packet and forwards the packet to the available SDN switches, while either making modifications in the flow-table entries or rerouting the packet. Otherwise, the controller also has a provision to drop the packet.

### 3.3 Probabilistic Analysis

We consider that SD-DCN comprises of a single controller unit and multiple OpenFlow switches. In this work, we focus on packet flow through an OpenFlow switch. Hence, we consider that the probability of a packet getting forwarded from the controller to the specific switch having queue length  $(Q + 1)$ , where indexing starts from the  $0^{th}$  position, is  $p'$ . If there are  $\mathcal{N}$  number of OpenFlow switches in the network, we consider that the probability of the packet getting forwarded to OpenFlow switch  $s$ , where  $1 \leq s \leq$

### 3. Theoretical Performance Analysis of SDN Switches

---

$\mathcal{N}$ , is defined as  $p_s$ . The packets get forwarded to anyone of the available OpenFlow switches without any preference. Therefore, the packet has an equal probability of getting forwarded to anyone of the OpenFlow switches. We have:

$$p_s = p' = \frac{1}{\mathcal{N}}, \quad \forall s \quad (3.1)$$

After getting forwarded by the controller, the packet gets queued at the switch buffer. Considering that the queue length is  $(Q + 1)$ , and the packet getting queued at any position of the buffer is unbiased, i.e., equally probable, we get:

$$P(B_i|C) = \frac{p'}{Q + 1}, \quad \text{where } 0 \leq i \leq Q \quad (3.2)$$

where  $B_i$  defines the  $i^{th}$  position of the buffer,  $C$  denotes the SDN controller, and  $P(B_i|C)$  denotes the probability of the packet getting forwarded to the buffer  $B_i$  from the controller  $C$ . After getting queued at buffer  $B_i$ , the packet gets forwarded to the next position of the OpenFlow queue,  $B_{i-1}$ , sequentially. Hence, we get:

$$P(B_{i-1}|B_i) = 1, \quad \text{where } 0 < i \leq Q \quad (3.3)$$

After reaching the  $0^{th}$  position of the queue, the packet enters the ingress processing unit of the OpenFlow switch. The packet is forwarded to the ingress flow-table having index 0, initially. Hence, the probability of the packet getting forwarded from the state  $B_0$  to the first ingress flow-table  $F_0$  is always unity. Mathematically,

$$P(F_0|B_0) = 1 \quad (3.4)$$

We consider that the packet is at the ingress flow-table  $F_i$ , the packet gets matched against the flow-table entries of  $F_i$ . The packet finds either table-hit or table-miss, as discussed in Section 3.2. We consider that the probability of having table-hit at the

### 3.3. Probabilistic Analysis

---

ingress flow-table  $F_i$  is  $p_i$ . Hence, the probability of table-miss at the ingress flow-table  $F_i$  is  $(1 - p_i)$ . Additionally, we consider that in case of table-hit at ingress flow-table  $F_i$ , where  $0 \leq i < n$ , there are three *equally probable* events, i.e., considered to be unbiased events, — (1) the packet gets forwarded to the egress flow-table  $F_e$ , (2) the packet is handled according to the output action, and (3) the packet gets forwarded to any of the next ingress flow-tables,  $F_j$ , where  $i < j \leq n$ . Hence, we get:

$$P(F_e|F_i) + P(O|F_i) + \sum_{j, n \geq j > i} P(F_j|F_i, \text{table-hit}) = p_i \quad (3.5)$$

where  $P(F_e|F_i)$ ,  $P(O|F_i)$ , and  $P(F_j|F_i, \text{table-hit})$  define the probability of the packet getting forwarded to the egress table  $F_e$ , the probability of packet executed according to the output action, and the probability of the packet getting forwarded to flow-table  $F_j$ , when there is table-hit, respectively. In case of table-miss at the ingress flow-table  $F_i$ , where  $0 \leq i < n$ , there are three *equally probable unbiased* events — (1) the packet gets forwarded to any of the next ingress flow-tables,  $F_j$ , where  $i < j \leq n$ , (2) the packet gets forwarded to the SDN controller, and (3) the packet gets dropped. We get:

$$\sum_{j, n \geq j > i} P(F_j|F_i, \text{table-miss}) + P(C|F_i) + P(D|F_i) = (1 - p_i) \quad (3.6)$$

where  $P(F_j|F_i, \text{table-miss})$ ,  $P(C|F_i)$ , and  $P(D|F_i)$  define the the probability of the packet getting forwarded to the flow-table  $F_j$ , when there is table-miss, the probability of packet getting forwarded to the controller, and the probability of the packet getting dropped, respectively. Therefore, the probability of the packet getting forwarded to a next ingress flow-table  $F_j$ , where  $i < j \leq n$ , i.e.,  $P(F_j|F_i)$ , where  $P(F_j|F_i) = P(F_j|F_i, \text{table-hit}) + P(F_j|F_i, \text{table-miss})$ , and the probability of the packet getting forwarded to the egress flow-table  $F_e$ , i.e.,  $P(F_e|F_i)$ , are as follows:

---

### 3. Theoretical Performance Analysis of SDN Switches

$$\begin{aligned}
 P(F_j|F_i) &= \frac{p_i}{3(n-i)} + \frac{1-p_i}{3(n-i)} \\
 &= \frac{1}{3(n-i)}, \quad 0 \leq i < n, \text{ and } j > i
 \end{aligned} \tag{3.7}$$

$$P(F_e|F_i) = \frac{p_i}{3}, \quad 0 \leq i < n \tag{3.8}$$

The packet cannot be forwarded to the ingress flow-table  $F_j$  from the ingress flow-table  $F_i$ . Hence, the probability of the aforementioned event is given as:

$$P(F_j|F_i) = 0, \quad 0 \leq i \leq n, \text{ and } j \leq i \tag{3.9}$$

If the packet finds a table-hit at the flow-table  $F_n$ , there are two *equally probable* events without biasness — (1) the packet get forwarded to the  $e^{th}$  egress flow-table,  $F_e$ , and (2) the packet is handled according to the output action. Therefore, the probability that the packet gets forwarded to the egress flow-table  $F_e$  from the ingress flow-table  $F_n$ , is given as:

$$P(F_e|F_n) = \frac{p_n}{2} \tag{3.10}$$

where  $p_n$  is the probability of getting a table-hit at the ingress flow-table  $F_n$ . Similar to the ingress flow-tables, at the egress flow-table  $F_{e+i}$ , we consider that the probability of getting a table-hit is defined as  $p_{e+i}$ , which depends on two *equally probable unbiased* outcomes — (1) the packet getting forwarded to any next egress flow-table  $F_{e+j}$ , and (2) the packet is handled according to the output action. On the other hand, in case of table-miss with probability  $(1 - p_{e+i})$ , which is also addition of three *equally probable* events, i.e., without having biasness, such as (1) the packet gets forwarded to any next egress flow-table,  $F_{e+j}$ , where  $i < j \leq m$ , (2) the packet gets forwarded to the SDN controller, and (3) the packet gets dropped. Hence, the probability of the packet getting forwarded to the next egress flow-table  $F_{e+j}$  from the egress flow-table  $F_{e+i}$  is given

### 3.3. Probabilistic Analysis

---

as  $P(F_{e+j}|F_{e+i}) = P(F_{e+j}|F_{e+i}, \text{table-hit}) + P(F_{e+j}|F_{e+i}, \text{table-miss})$ . Therefore,  $P(F_{e+j}|F_{e+i})$  is equated as follows:

$$\begin{aligned} P(F_{e+j}|F_{e+i}) &= \frac{p_{e+i}}{2(m-i)} + \frac{1-p_{e+i}}{3(m-i)} \\ &= \frac{2+p_{e+i}}{6(m-i)}, \quad 0 \leq i < m, \text{ and } j > i \end{aligned} \quad (3.11)$$

On the other hand, similar to the ingress flow-table rules, the packet cannot flow to any egress flow-table with lower index. Therefore, the probability of the packet getting forwarded to the egress flow-table  $F_{e+j}$  to the egress flow-table  $F_{e+i}$ , where  $j \leq i$ , is given as follows:

$$P(F_{e+j}|F_{e+i}) = 0, \quad 0 \leq i \leq m, \text{ and } j \leq i \quad (3.12)$$

From Figure 3.1, we get that the packet may reach to the *output action* state, denoted as  $O$ , from the states — any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ , and any of the egress flow-table,  $F_{e+i}$ , where  $0 \leq i \leq m$ , when there is a table-hit. We define the probability,  $P(O|F_i)$ , of the packet reaching to output action state from any flow-table  $F_i$ , i.e., either ingress or egress flow-table, is given as follows:

$$P(O|F_i) = \begin{cases} \frac{p_i}{3}, & \text{if } 0 \leq i < n \\ \frac{p_i}{2}, & \text{if } i = n \\ \frac{p_{e+i}}{2}, & \text{if } 0 \leq i < m \\ p_{e+i}, & \text{if } i = m \end{cases} \quad (3.13)$$

On the other hand, from Figure 3.1, we observe that the packet may get dropped, where the state represented as  $D$ , from the states such as any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ , and any of the egress flow-tables,  $F_{e+i}$ , where  $0 \leq i \leq m$ , when there is a table-miss. We define the probability,  $P(D|F_i)$ , of the packet reaching to packet drop state from any flow-table  $F_i$  is given as follows:

### 3. Theoretical Performance Analysis of SDN Switches

---

$$P(D|F_i) = \begin{cases} \frac{1-p_i}{3}, & \text{if } 0 \leq i < n \\ \frac{1-p_i}{2}, & \text{if } i = n \\ \frac{1-p_{e+i}}{3}, & \text{if } 0 \leq i < m \\ \frac{1-p_{e+i}}{2}, & \text{if } i = m \end{cases} \quad (3.14)$$

Similarly, from Figure 3.1, we get that the packet may get forwarded to the SDN controller, where the state represented as  $C$ , from the states such as any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ , and any of the egress flow-tables,  $F_{e+i}$ , where  $0 \leq i \leq m$ , when there is a table-miss. We define the probability,  $P(C|F_i)$ , of the packet reaching to packet drop state from any flow-table  $F_i$  is given as follows:

$$P(C|F_i) = \begin{cases} \frac{1-p_i}{3}, & \text{if } 0 \leq i < n \\ \frac{1-p_i}{2}, & \text{if } i = n \\ \frac{1-p_{e+i}}{3}, & \text{if } 0 \leq i < m \\ \frac{1-p_{e+i}}{2}, & \text{if } i = m \end{cases} \quad (3.15)$$

Based on aforementioned state transition conditional probability, we calculate the probability of the packet to be any of the aforementioned state. Based on Equations (3.2) and (3.3), probability of the packet to be at the  $i^{\text{th}}$  position of the queue,  $B_i$ , where  $0 \leq i \leq Q$ , i.e.,  $P(B_i)$ , is defined as follows:

$$\begin{aligned} P(B_i) &= P(B_i|C)P(C) + P(B_i|B_{i+1})P(B_{i+1}) \\ &= \sum_{j=i}^Q \frac{p'}{Q+1} = p' \left(1 - \frac{i}{Q+1}\right), \quad 0 \leq i \leq Q \end{aligned} \quad (3.16)$$

where  $P(C)$  defines the probability of packet to be at SDN controller. We consider that  $P(C) = 1$ . The probability of packet entering the ingress flow-table  $F_0$ ,  $P(F_0)$ , is calculated as follows, based on Equations (3.4) and (3.16):

$$P(F_0) = P(F_0|B_0)P(B_0) = p' \quad (3.17)$$



### 3.3. Probabilistic Analysis

---

The packet reaches the ingress flow-table  $F_1$ , if and only if the packet gets forwarded by the ingress flow-table  $F_0$ . Hence, from Equations (3.7) and (3.17), the probability of packet being in the ingress flow-table  $F_1$ ,  $P(F_1)$ , is defined as follows:

$$P(F_1) = P(F_1|F_0)P(F_0) = \frac{p'}{3n} \quad (3.18)$$

Additionally, from Equations (3.7), (3.17), and (3.18), the probability of the packet being in the ingress flow-table  $F_i$ ,  $P(F_i)$ , where  $0 < i \leq n$ , is calculated as follows:

$$\begin{aligned} P(F_i) &= \sum_{j=0}^{i-1} P(F_i|F_j)P(F_j) \\ &= \frac{p'}{3n} \prod_{j=1}^{i-1} \left[ 1 + \frac{1}{3(n-j)} \right], \quad 1 < i \leq n \end{aligned} \quad (3.19)$$

From Figure (3.1), we observe that the packet can reach to egress flow-table  $F_e$  from any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ . Hence, based on Equations (3.8), (3.10), and (3.19), the probability of the packet to be at the egress flow-table  $F_e$ ,  $P(F_e)$ , is defined as in Equation (3.20).

$$\begin{aligned} P(F_e) &= \sum_{j=0}^n P(F_e|F_j)P(F_j) \\ &= \frac{p'}{3} \left( \frac{p_1}{3n} + p_0 \right) \sum_{i=2}^{n-1} \left( \frac{p_i}{3} \frac{p'}{3n} \prod_{j=1}^{i-1} \left[ 1 + \frac{1}{3(n-j)} \right] \right) + \frac{p_n}{2} \frac{p'}{3n} \prod_{j=1}^{n-1} \left[ 1 + \frac{1}{3(n-j)} \right] \end{aligned} \quad (3.20)$$

Additionally, using *Stirling's approximation* formula on Equation (3.20), we get the upper and lower bound of  $P(F_e)$  as defined in Equation (3.21).

$$\left[ \frac{1}{9n} pp' + \frac{10}{27} pp' - \frac{1}{54n} pp' \ln n \right] < P(F_e) < \left[ \frac{1}{9n} pp' \ln n + \frac{7}{18} pp' + \frac{1}{9n} pp' \right] \quad (3.21)$$

From Figure 3.1, we observe that the packet can only reach the egress flow-table  $F_{e+1}$  from the egress flow-table  $F_e$ . Hence, using Equations (3.11) and (3.20), we get

---

### 3. Theoretical Performance Analysis of SDN Switches

---

the probability of the packet to be in egress flow-table  $F_{e+1}$ ,  $P(F_{e+1})$ , is as follows:

$$\begin{aligned} P(F_{e+1}) &= P(F_{e+1}|F_e)P(F_e) \\ &= \left(\frac{2+p}{6m}\right) P(F_e) \end{aligned} \quad (3.22)$$

Using Equations (3.11), (3.20), and (3.22), we define the probability of the packet being at egress flow  $F_{e+i}$ ,  $P(F_{e+i})$ , where  $1 < i \leq m$ , as follows:

$$\begin{aligned} P(F_{e+i}) &= \sum_{j=0}^{i-1} P(F_{e+i}|F_{e+j})P(F_{e+j}) \\ &= \frac{(2+p)}{6m} P(F_e) \prod_{j=1}^{i-1} \left[1 + \frac{2+p}{6(m-j)}\right], \quad 1 < i \leq m \end{aligned} \quad (3.23)$$

Using Stirling's approximation formula on Equation (3.23), we get the upper and lower bounds of  $P(F_{e+i})$ , as follows:

$$\frac{(2+p)^2}{36m} P(F_e) \ln \left(\frac{m}{m-i+1}\right) < P(F_{e+i}) \leq \frac{(2+p)}{6m} P(F_e) \left[\frac{m}{m-i+1}\right]^{\left(\frac{2+p}{6}\right)} \quad (3.24)$$

#### 3.3.1 Output Action Probability

As shown in Figure 3.1, the packet reaches the output action state from either any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ , or any of the egress flow-tables,  $F_{e+j}$ , where  $0 \leq j \leq m$ . Hence, the probability of the packet being in the output action state,  $P(O)$ , depends on Equations (3.11), (3.13), (3.17)–(3.20), (3.22), and (3.23). We define  $P(O)$  as shown in Equation (3.25).

$$\begin{aligned} P(O) &= \sum_{i=0}^n P(O|F_i)P(F_i) + \sum_{j=0}^m P(O|F_{e+j})P(F_{e+j}) \\ &= \frac{p}{3} \left[ P(F_0) + P(F_1) + \sum_{i=2}^{n-1} P(F_i) \right] + \frac{p}{2} \left[ P(F_n) + P(F_e) + P(F_{e+1}) + \sum_{j=2}^{m-1} P(F_{e+j}) \right] \\ &\quad + pP(F_{e+m}) \end{aligned} \quad (3.25)$$

### 3.3. Probabilistic Analysis

---

Additionally, using Stirling's approximation formula on Equation (3.25), we get the upper and lower bounds of  $P(O)$ , as given in Equation (3.26).

$$\begin{aligned}
& \left[ \frac{1}{3} pp' \left( 1 + \frac{1}{3n} + \frac{1}{9n} \ln n \right) + \frac{p}{2} \left[ \frac{p'}{9n} \ln n + P(F_e) + \left( \frac{2+p}{6m} \right) P(F_e) + \left( \frac{(2+p)^2}{36m} \right) P(F_e) \right. \right. \\
& \left. \left. ((m-2) \ln m - (m-1) \ln(m-1) - (m-1)) \right] + p \left( \frac{(2+p)^2}{36m} \right) P(F_e) \ln m \right] < P(O) < \\
& \left[ \frac{1}{3} pp' \left( 1 + \frac{1}{3n} + \frac{1}{3} (\ln n - 1) \right) + \frac{p}{2} \left[ \frac{p'}{9n} \ln n + P(F_e) + \left( \frac{2+p}{6m} \right) P(F_e) + \left( \frac{(2+p)}{6} \right) P(F_e) \right. \right. \\
& \left. \left. (\ln m - 1) \right] + p \left( \frac{(2+p)^2}{6} \right) P(F_e) \ln m \right] \quad (3.26)
\end{aligned}$$

#### 3.3.2 Packet Drop Probability

From Figure 3.1, we observe that the packet reaches the packet drop state,  $D$ , from either any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ , or any of the egress flow-tables,  $F_{e+j}$ , where  $0 \leq j \leq m$ . Hence, the probability of the packet getting dropped depends on Equations (3.11), (3.14), (3.17)–(3.20), (3.22), and (3.23). We define  $P(D)$  as given in Equation (3.27).

$$\begin{aligned}
P(D) &= \sum_{i=0}^n P(D|F_i)P(F_i) + \sum_{j=0}^m P(D|F_{e+j})P(F_{e+j}) \\
&= \left( \frac{1-p}{3} \right) \left[ P(F_0) + P(F_1) + \sum_{i=2}^{n-1} P(F_i) + P(F_e) + P(F_{e+1}) + \sum_{j=2}^{m-1} P(F_j) \right] + \\
& \quad \left( \frac{1-p}{2} \right) [P(F_n) + P(F_{e+m})] \quad (3.27)
\end{aligned}$$

Additionally, using Stirling's approximation formula on Equation (3.27), we get the upper and lower bounds of  $P(D)$ , as shown in Equation (3.28).

$$\begin{aligned}
& \left[ \left( \frac{1-p}{3} \right) p' \left( 1 + \frac{1}{3n} + \frac{1}{9n} \ln n \right) + \left( \frac{1-p}{3} \right) P(F_e) \left[ 1 + \left( \frac{2+p}{6m} \right) + \left( \frac{(2+p)^2}{36m} \right) \right. \right. \\
& \left. \left. ((m-2) \ln m - (m-1) \ln(m-1) - (m-1)) \right] + \left( \frac{1-p}{2} \right) \left( \frac{p'}{9n} \ln n + \left( \frac{(2+p)^2}{36m} \right) \right. \right. \\
& \left. \left. P(F_e) \ln m \right] \right] < P(D) < \left[ \left( \frac{1-p}{3} \right) p' \left( 1 + \frac{1}{3n} + \frac{1}{3} (\ln n - 1) \right) + \left( \frac{1-p}{3} \right) P(F_e) \left[ 1 + \right. \right. \\
& \left. \left. \left( \frac{2+p}{6m} \right) + \left( \frac{2+p}{6} \right) (\ln m - 1) \right] + \left( \frac{1-p}{2} \right) \left( \frac{p'}{9n} \ln n + \left( \frac{(2+p)^2}{36m} \right) P(F_e) \ln m \right) \right]
\end{aligned} \tag{3.28}$$

### 3.3.3 Send to Controller Probability

From Figure 3.1, we get that the packet reaches the controller from either any of the ingress flow-tables,  $F_i$ , where  $0 \leq i \leq n$ , or any of the egress flow-tables,  $F_{e+j}$ , where  $0 \leq j \leq m$ . Hence, the probability of the packet getting forwarded to the controller depends on Equations (3.11), (3.15), (3.17)–(3.20), (3.22), and (3.23). We define  $P(C)$  as follows:

$$P(C) = \sum_{j=0}^N P(C|F_j)P(F_j) + \sum_{k=0}^M P(C|F_{e+k})P(F_{e+k}) \tag{3.29}$$

which is same as  $P(D)$ .

## 3.4 Performance Analysis

In this Section, using AMOPE, we analyze the performance of packet flow through an OpenFlow switch in SD-DCN. We evaluate the performance of AMOPE based on the parameters mentioned in Section 3.4.2. We simulated AMOPE in the MATLAB simulation platform. For simplicity, we consider that the number of OpenFlow switch in SD-DCN is two, i.e.,  $p' = \frac{1}{2}$ , where  $p'$  is the probability of the packet getting forwarded to the concerned switch. Additionally, we consider that if a packet is forwarded to the

### 3.4. Performance Analysis

---

controller, it eventually, is queued in an OpenFlow switch.

#### 3.4.1 Simulation Parameters

In AMOPE, simulations are performed for the OpenFlow switch in SD-DCN with a single controller and two OpenFlow switches. We considered that the packet arrival rate and the packet service rate per OpenFlow switch are approximately 0.2 million packets per second (mpps) [9] and 0.03 mpps [87], respectively. We considered different simulation parameters, as shown in Table 3.1. The simulation time is 5 sec, queue size per OpenFlow switch is 0.73 million packets [9]. We considered that there are 10 number of ingress flow-tables and either zero or 10 number of egress flow-tables, as shown in Table 3.1.

**Table 3.1:** Simulation parameters

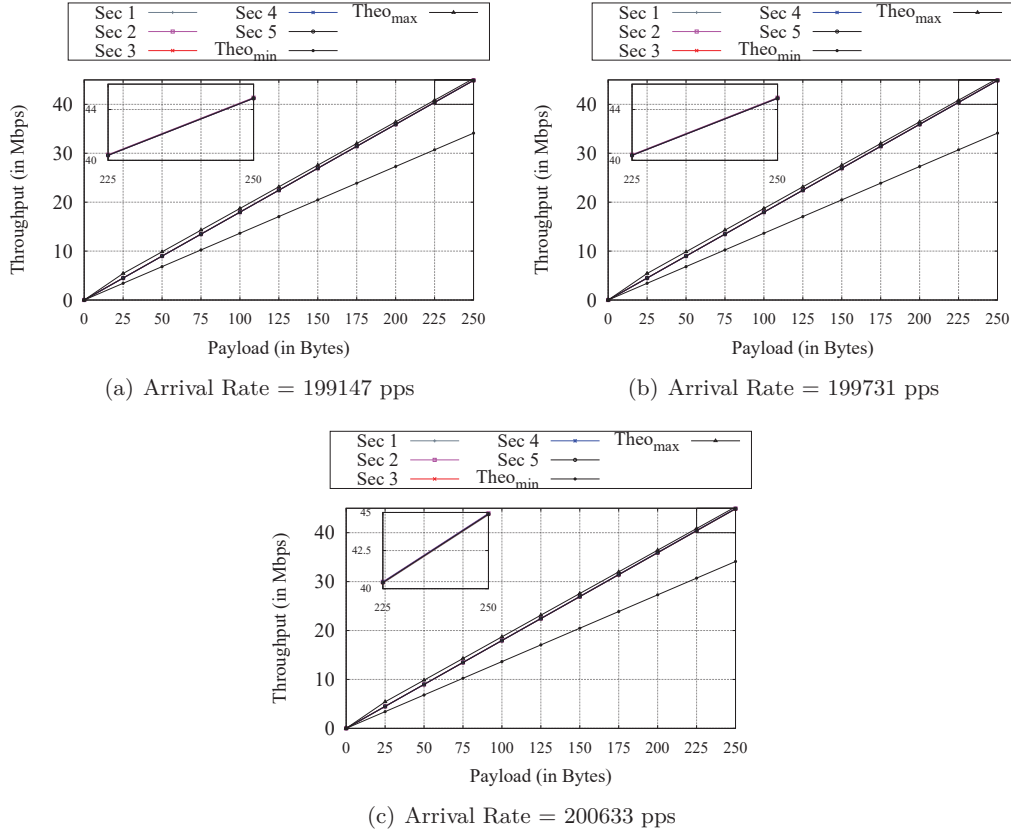
Parameter	Value
Number of OpenFlow switch	2
Packet arrival rate per switch	0.199147, 0.199731, 0.200633 mpps
Packet service rate per switch	0.03 mpps [87]
Queue size per switch	0.73 million packets [9]
Flow table lookup time	33.33333 $\mu$ sec [87]
Number of ingress tables	10
Number of egress tables	{0, 10}

#### 3.4.2 Performance Metrics

We evaluate the performance of the OpenFlow switch based on the Markov chain-based analytical model with different packet arrival rates — 0.199147, 0.199731, 0.200633 million packets per second (mpps), while considering the following parameters:

**Throughput:** We consider that the throughput of an OpenFlow switch is defined as the number of packets processed, i.e., reaches the output action state. A packet can reach the output action state from any ingress or egress flow-tables.

### 3. Theoretical Performance Analysis of SDN Switches

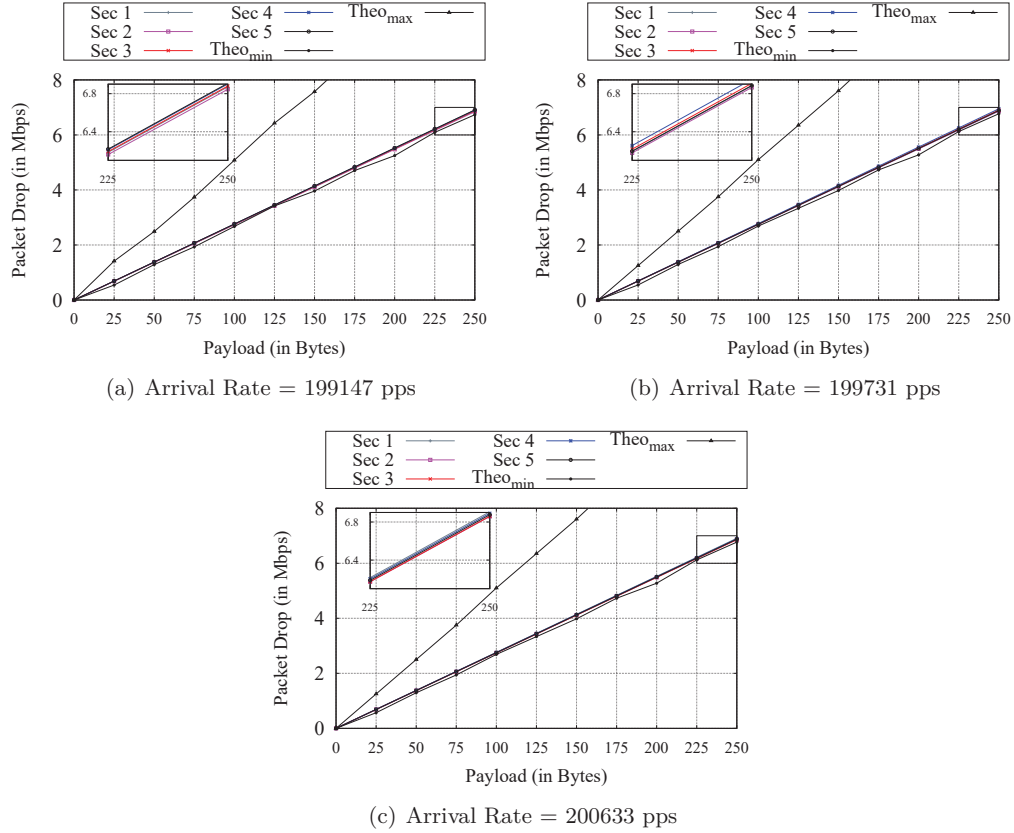


**Figure 3.3:** Sent to Output Rate of an OpenFlow Switch

**Number of Packets Dropped:** A packet can be dropped due to following reasons — there is no table-miss entry for ingress and egress flow-table or the table-miss entry is to drop the packet, the output action is not defined for matched entry at ingress or egress flow-table, and the action specified by the table-miss flow entry is drop.

**Number of Packets Sent to Controller:** A packet is sent to the controller if the action mentioned in the table-miss entry is to forward a packet to the controller. The packets, which are forwarded to the controller from the OpenFlow switches, are considered to be queued again in one of the available OpenFlow switches.

### 3.4. Performance Analysis



**Figure 3.4:** Packet Drop Rate of an OpenFlow Switch

**Average Queuing Packet Delay:** We calculate the average queuing packet delay as the duration between timestamp when a packet enters into OpenFlow switch, and the time stamp when the packet enters through ingress port for processing.

**Packet Processing Time:** We consider the packet process time is the duration between the time stamp when a packet enters to ingress flow-table  $F_0$  and the time stamp when the packet gets out of the switch.

### 3. Theoretical Performance Analysis of SDN Switches

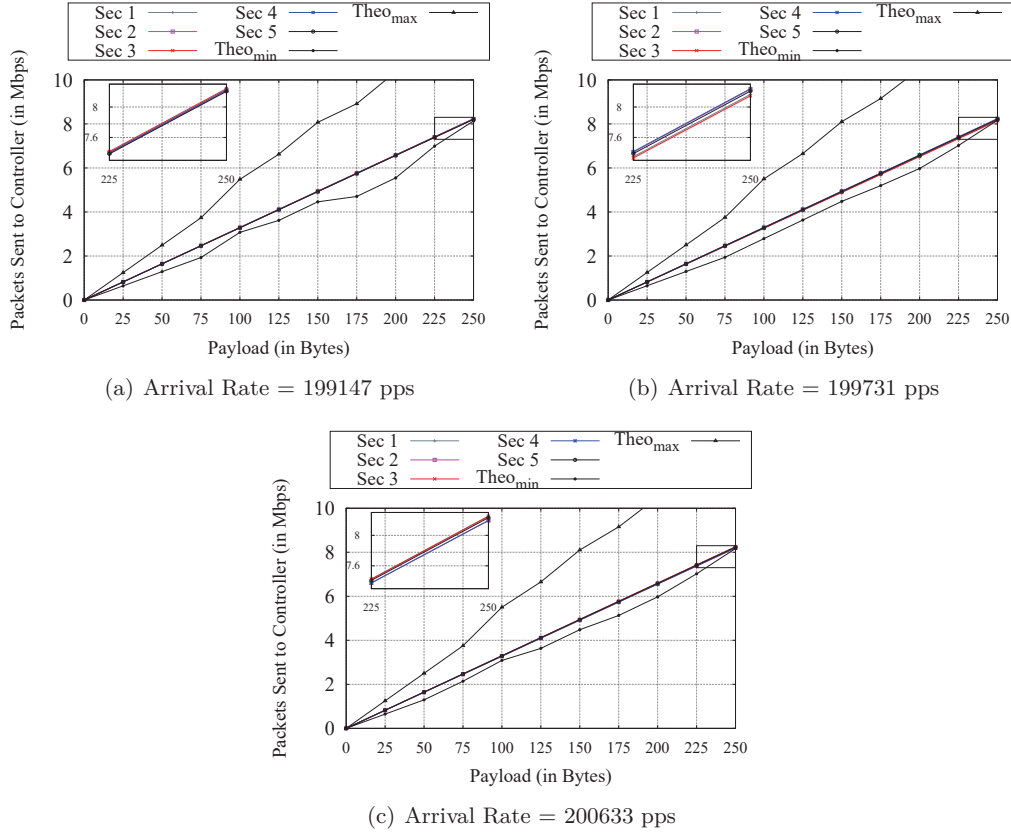


Figure 3.5: Send to Controller Rate of an OpenFlow Switch

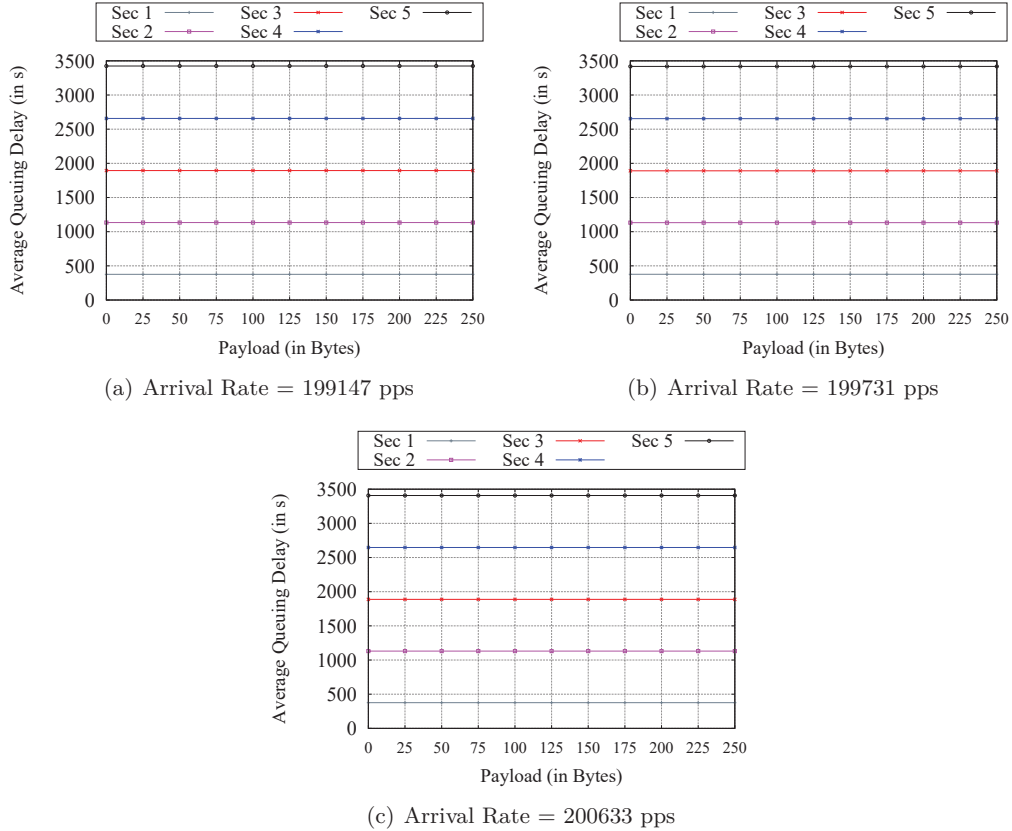
#### 3.4.3 Result and Discussion

For simulation, we generated random numbers from the *Poisson distribution* with a mean packet arrival rate of 0.2 mpps, as we considered the Markovian process. Additionally, we considered randomness, while taking a decision on table-hit and table-miss. If there is table-miss flow entry, and action mentioned for table-miss flow entry is to forward to the controller, the packets get queued again in the OpenFlow switch buffer.

From Figure 3.3, we observe that approximately 9% of the arrived number of packets are sent for output action. In Figures 3.3(a), 3.3(b), and 3.3(c), the throughput of an OpenFlow switch increases with the increase in payload size. Additionally, we get that



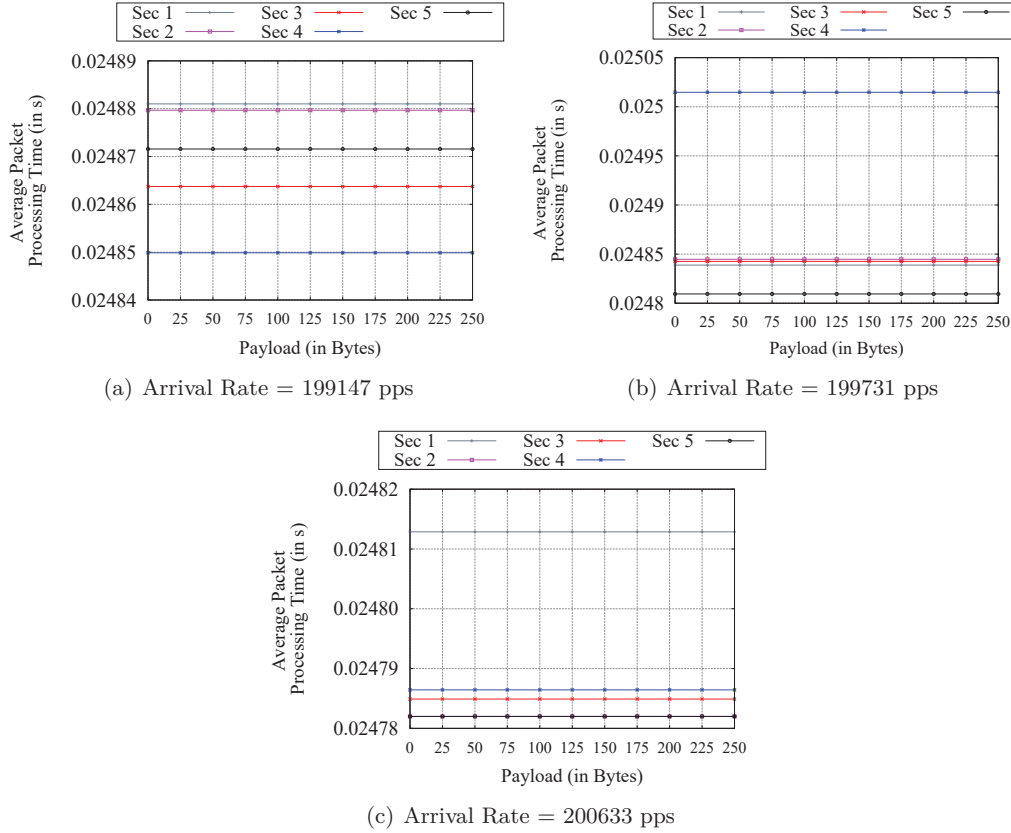
### 3.4. Performance Analysis



**Figure 3.6:** Average Queuing Delay of an OpenFlow Switch

the throughput in each second is almost similar. Hence, we conclude that the throughput of an OpenFlow switch depends on the payload as well as on the number of matched packets. On the other hand, Figure 3.4 shows that almost 60% of the packets are dropped, as there is either no table-miss flow entry for ingress and egress of flow-tables, or the action mentioned in the table-miss flow entry is packet drop, or any output action is not mentioned in the matched entry. Here, from Figures 3.4(a), 3.4(b), and 3.4(c), we observe that the packet drop (in *Mbps*) increases with the increase in payload. Additionally, we conclude that OpenFlow considers each packet as an individual entity, and process separately. Additionally, from Figures 3.5(a), 3.5(b), and 3.5(c), we yield that the approximately 31% of the arrived packets are sent to the controller, and sent

### 3. Theoretical Performance Analysis of SDN Switches



**Figure 3.7:** Average Packet Processing Delay of an OpenFlow Switch

back to OpenFlow switch queue again. From Figures 3.3, 3.4, and 3.5, we observe that the simulated results lie within the theoretical minimum and maximum values obtained using AMOPE.

Figures 3.6 and 3.7 depict two types of delay of the OpenFlow switch such as queuing and processing delay. From Figures 3.6(a), 3.6(b), and 3.6(c), we observe that the average queuing delay is much higher compared to processing delay. Hence, we conclude that the packet delay at the OpenFlow switch increases mostly due to the packet queuing delay. On the other hand, from Figures 3.7(a), 3.7(b), and 3.7(c), we observe that the average packet processing time is almost similar for each time instant. For each time instant, the packet processing delay is in the range  $[33.3333 \mu\text{sec} - 0.025 \text{ sec}]$ . We get

### 3.5. Concluding Remarks

---

approximately 0.02 sec processing delay, in case of the packet has to go through for match entries for each ingress and egress flow-tables. In Figures 3.7(a), 3.7(b), and 3.7(c), we see that the average processing delay varies randomly for different arrival rate, as the packet processing delay solely depends on the number of flow-tables the packet has to go through for finding a match. Additionally, from Figures 3.6 and 3.7, we get that delay factor are almost linear with the variation of payload, as the processing time in an OpenFlow switch depends on the header size of the packet, i.e., the matched field entries, and does not depends on the payload.

We observe that the packet delay can be improved, while using an efficient queuing algorithm for an OpenFlow switch. On the other hand, the packet drop rate is too high for an OpenFlow switch due to limitations of TCAM memory size, and the mismatch of rules. Hence, we suggest that the packet drop rate can be improved, while using TCAM memory, efficiently, and using a proper rule placement mechanism.

### 3.5 Concluding Remarks

In this Chapter, we analyzed the performance of packet flow through an OpenFlow switch in SD-DCNs and proposed AMOPE, an analytical model, to define the probabilistic bounds of the performance metrics of the OpenFlow switch. We modeled the packet flow steps in an OpenFlow switch using Markov chain and calculated the theoretical probabilities of the packet to be any state. Additionally, we calculated the probabilities of a packet being at output action state, packet getting dropped, and packet getting forwarded to the controller, theoretically. We also verified the theoretical findings using the MATLAB simulation platform. Simulation-based analysis exhibited that approximately 60% of the processed packets are sent to output action, 31% of the processed packets are sent to the controller, and the remaining processed packets are dropped in an OpenFlow switch. We inferred that in an OpenFlow switch, the total delay is high due to a high delay at the queue of the OpenFlow switch. On the other hand, in an OpenFlow switch,

### **3. Theoretical Performance Analysis of SDN Switches**

---

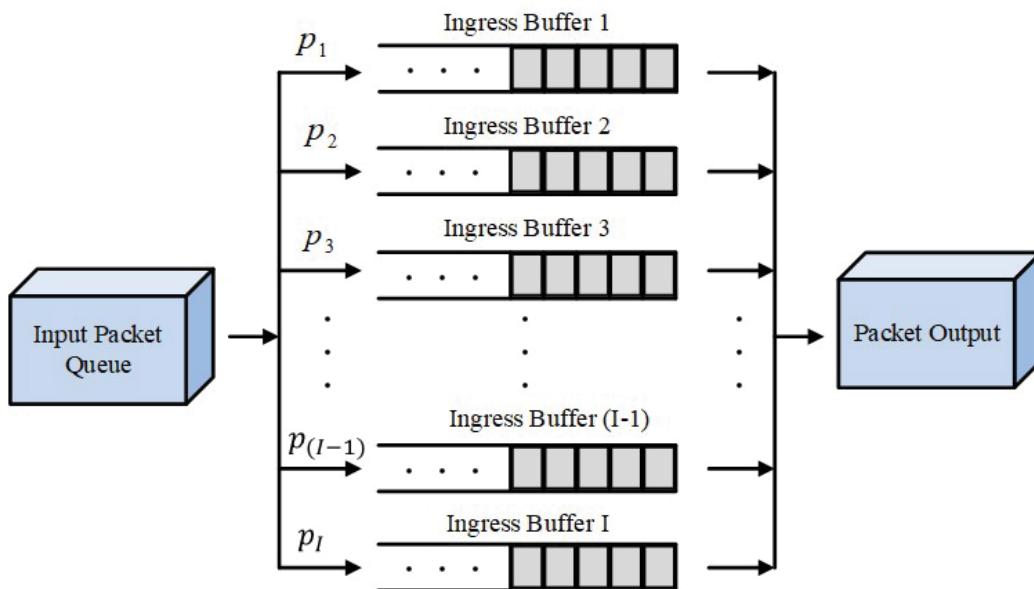
a high number of packets get dropped due to either not having table-miss flow entry, or output action not being specified.

## Chapter 4

# Buffer Size Analysis of SDN Switches

In this Chapter, we propose an analytical scheme, named OPUS, for buffer bound evaluation of an OpenFlow system. Additionally, we propose a queueing scheme for an OpenFlow system — I-M/M/1/K queueing model — based on the OpenFlow specification version 1.5.0. Further, we calculate the minimum buffer size requirement of an OpenFlow switch, theoretically.

This Chapter is organized as follows. The design of OPUS is proposed in Section 4.1. Section 4.2 focuses on the formulation of the proposed analytical model, OPUS. Additionally, we evaluate two cases in Section 4.3. Thereafter, Section 4.4 discusses the performance evaluation of the required buffer size of an OpenFlow switch-based system with varying packet arrival and processing rate in SD-DCN. Finally, Section 4.5 concludes this Chapter.

Figure 4.1: OpenFlow Switch with  $I$  Ingress Ports/Buffers

## 4.1 System Model

In this section, we present the architecture of an OpenFlow switch-based system in SD-DCNs. According to the OpenFlow specification version 1.5.0 [5], in an OpenFlow switch, there exist multiple ingress and output ports. Each incoming packet is directed to an ingress port based on the port number embedded in the packet. We consider that there are  $I$  ingress ports in an OpenFlow switch, as shown in Figure 4.1. Each ingress port  $i \in [1, I]$  has a fixed size of buffer, which is denoted as  $K_i$ , as shown in Figure 4.2. At ingress port  $i$ , the mean packet arrival rate and the mean packet processing rate are denoted as  $\lambda_i$  and  $\mu_i$ , respectively. Hence, the traffic intensity of buffer  $i$  in an OpenFlow switch is defined as  $\frac{\lambda_i}{\mu_i}$ . The packets from each buffer are processed against the same set of flow-rules. According to the OpenFlow specification version 1.5.0, multiple flow-tables exist in an OpenFlow switch. After reaching an OpenFlow switch, each packet gets forwarded to one of the available ingress ports, e.g., ingress port  $i$ . Thereafter, it gets

## 4.1. System Model

---

queued in the buffer of the ingress port  $i$ , i.e., queued at state  $b_{m,i}$ , where  $m \in [0, K_i)$ .

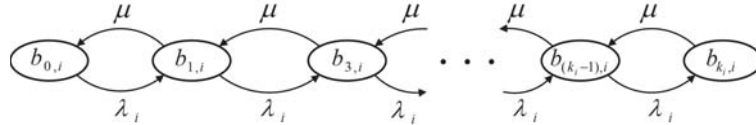
We consider that packet processing at an OpenFlow switch follows *Markovian Process*.

### 4.1.1 Markovian Process: The Justification

We studied the behavior of an OpenFlow switch using *Markovian model* [88], as it follows the following Markov properties:

1. Each packet is processed individually, and the behavior of an OpenFlow switch is *memoryless*.
2. Packet processing in an OpenFlow switch is a stochastic process having Markov properties, as the conditional probability distribution of the future state depends only on the present state, not on the series of states followed in the past.

We consider that a packet gets queued at state  $X_0$ , and follows the sequence  $X_0 \rightarrow X_1 \rightarrow \dots \rightarrow X_t$ , where  $X_t$  defines the state of the packet at time instant  $t$ . Therefore, at time instant  $(t + 1)$ , the probability of the packet to be in state  $X_{t+1}$  is defined as  $P[X_{t+1}|X_t, X_{t-1}, \dots, X_2, X_1, X_0] = P[X_{t+1}|X_t]$ , where  $X_t$ ,  $X_{t-1}$ , and  $X_{t+1}$  are the present, immediate past, and future state of a Markovian process, respectively. We consider that the packet arrival rate and the time between arrivals follow Poisson distribution and exponential distribution, respectively, as given in Theorem 4.1.



**Figure 4.2:** The Ingress Port/Buffer  $i$  of an OpenFlow Switch

**Theorem 4.1.** *Considering that the packet processing at an OpenFlow switch follows the Markovian process, the arrival of packets and the time between arrivals follow the Poisson distribution and exponential distribution, respectively.*

*Proof.* Motivated by *Chapman-Kolmogorov* dynamics [88], we consider that in an infinitesimal time duration  $(t, t + \Delta t)$ , the mutually exclusive and exhaustive events may occur such as (1) One packet arrives to the buffer of an OpenFlow switch, (2) one packet gets processed and no packet arrival in an OpenFlow switch, and (3) the number packets in the buffer remains same. Based on these events, the rate of change in packet flow,  $\frac{dq_{m,i}}{dt}$ , at  $m^{th}$  state of buffer  $i$  is defined as follows:

$$\frac{dq_{m,i}(t)}{dt} = \begin{cases} -(\lambda_{m,i} + \mu_{m,i})q_{m,i}(t) + \lambda_{(m-1),i}q_{(m-1),i}(t) \\ \quad + \mu_{(m+1),i}q_{(m+1),i}(t), & \text{if } m \geq 1 \\ -\lambda_{0,i}q_{0,i}(t) + \mu_{1,i}q_{1,i}(t), & \text{if } m = 0 \end{cases} \quad (4.1)$$

By solving Equation (4.1), we get that the probability of packet getting queued at the ingress buffer of an OpenFlow switch follows the Poisson distribution. On the other hand, the time between arrivals follows the exponential distribution.  $\square$

#### 4.1.2 Packet Flow through an OpenFlow Switch

Initially, the incoming packets get queued at the ingress buffers. Thereafter, from  $b_{0,i}$  of the buffer, each packet enters the ingress flow-table for rule matching, as mentioned in OpenFlow version 1.5.0 [5]. If there is a table hit, then the packet follows the action mentioned in the corresponding matched rule. The action can be one of these — (1) the packet goes to another ingress flow-table having higher index than the current index of the ingress flow-table; (2) the packet enters an egress flow-table if the egress flag of the packet is set; and (3) the packet goes to the output port or gets dropped, according to the action mentioned in the matched flow rule.

In case of table miss, the action can be one of these — (1) the packet gets forwarded to the next flow-table; (2) the packets get forwarded to the controller, according to the table-miss entry; and (3) the packet may get dropped, if either the action in the miss flow entry is to drop the packet, or there is no table-miss entry.



## 4.2. OPUS Scheme: I-M/M/1/K Queue

---

We consider that, on an average, each packet gets processed by an OpenFlow switch in  $\frac{1}{\mu_i}$  time units. Based on this observation, we model the buffer at each ingress port of an OpenFlow switch as the M/M/1/K model. Additionally, we consider that there are  $I$  ingress ports in an OpenFlow switch. Therefore, in OPUS, we present the queueing model in an OpenFlow switch as I-M/M/1/K Queueing Model, which is discussed in Section 4.2.

## 4.2 OPUS Scheme: I-M/M/1/K Queue

In OPUS, we consider that there are  $I$  number of ingress ports. In other words, there is  $I$  number of ingress buffers. Each packet can be queued at any of the  $I$  ingress buffers based on the ingress port mentioned in the packet. We consider that a packet can be forwarded to the ingress buffer  $i$  with probability  $q_i$ . Therefore, we get  $\sum_{i=1}^I q_i = 1$ .

We consider that  $K_i$  denotes the size of the buffer  $i$ . We define each position in the buffer as a state. Therefore, the  $m^{th}$  state of buffer  $i$  means the  $m^{th}$  position of buffer  $i$ . At any infinitesimal time interval  $dt$ , the packets can enter the state  $m$  of buffer  $i$  in two distinct events — (1) New packets are added to the queue at the arrival rate of  $\lambda_{(m-1),i}$ , i.e., state transition follows  $[b_{(m-1),i} \rightarrow b_{m,i}]$  at a rate of  $\lambda_{(m-1),i}$ ; and (2) Queued packets get processed by the system at a rate of  $\mu_{(m+1),i}$ , i.e., state transition follows  $[b_{(m+1),i} \rightarrow b_{m,i}]$  at a rate of  $\mu_{(m+1),i}$ . Therefore, in  $dt$  time interval, the incoming packet flow,  $\text{IP}_{m,i}$ , in the  $m^{th}$  state of buffer  $i$  is defined as —  $\text{IP}_{m,i} = q_{(m-1),i}\lambda_{(m-1),i}dt + q_{(m+1),i}\mu_{(m+1),i}dt$ .

On the other hand, at infinitesimal time interval  $dt$ , the packets can leave from the state  $m$  of buffer  $i$  in two distinct events — (1) new packets are added to the queue at the arrival rate of  $\lambda_{m,i}$ . State transition follows  $[b_{m,i} \rightarrow b_{(m+1),i}]$  at the rate of  $\lambda_{m,i}$ , and (2) queued packets get processed by the system at the rate of  $\mu_{m,i}$ , i.e., state transition follows  $[b_{m,i} \rightarrow b_{(m-1),i}]$  at a rate of  $\mu_{m,i}$ . Therefore, outgoing packet flow,  $\text{OP}_{m,i}$ , in  $dt$  time interval from  $m^{th}$  state of buffer  $i$  is defined as —  $\text{OP}_{m,i} = q_{m,i}\lambda_{m,i}dt + q_{m,i}\mu_{m,i}dt$ . Hence, considering that  $\frac{dq_{m,i}}{dt} = 0$ , we get —  $g_{(m-1),i} = g_{m,i} = \text{constant}$ , where  $g_{m,i} =$

---

#### 4. Buffer Size Analysis of SDN Switches

$q_{m,i}\lambda_{m,i} - q_{(m+1),i}\mu_{(m+1),i}$ . Therefore, from Equation (4.1), we get:

$$q_{m,i} = q_{0,i} \prod_{j=0}^{m-1} \frac{\lambda_{j,i}}{\mu_{(j+1),i}}, \quad \forall m \in (0, K_i] \quad (4.2)$$

As per OpenFlow specification version 1.5.0 [5], the packets from different ingress buffers get matched against the flow-table entries, simultaneously. Therefore, the packet process rate of the system is fixed for an OpenFlow switch-based system. The packet processing rate of buffer  $i$  is denoted as follows:

$$\mu_{m,i} = \mu, \quad \forall m \in [0, K_i] \text{ and } \forall i \in [1, I] \quad (4.3)$$

where  $\mu$  is a constant for an OpenFlow switch-based system. On the other hand, the packet arrival rate varies for each buffer  $i$ . Therefore, the packet arrival rate for buffer  $i$  is denoted as follows:

$$\lambda_{m,i} = \begin{cases} \lambda_i & \forall m \in [0, K_i] \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

where  $\lambda_i$  is the packet arrival rate for buffer  $i$ . Additionally, according to the *Poisson's splitting rule* [89], we get  $\sum_{i=1}^I \lambda_i = \lambda$  and  $\lambda_i = q_i \lambda$ . Based on Equations (4.3) and (4.4), the probability of a arrived packet to be at the  $m^{\text{th}}$  state of buffer  $i$ ,  $q_{m,i}$ , is re-defined as follows:

$$q_{m,i} = q_{0,i} \prod_{j=0}^{m-1} \frac{\lambda_i}{\mu} = q_{0,i} \left(\frac{\lambda_i}{\mu}\right)^m \quad (4.5)$$

Therefore, considering  $\sum_{m=0}^{K_i} q_{m,i} = q_i$ , from Equation (4.5), we evaluate the probability of an arrived packet to be in the  $0^{\text{th}}$  state of buffer  $i$ , i.e.,  $q_{0,i}$ , as follows:

$$q_{0,i} = q_i \left[ \frac{1 - \frac{\lambda_i}{\mu}}{1 - \left(\frac{\lambda_i}{\mu}\right)^{K_i+1}} \right] \quad (4.6)$$

## 4.2. OPUS Scheme: I-M/M/1/K Queue

---

Using OPUS, we measure the performance of an OpenFlow switch-based system based on the parameters — (1) expected number of packets in the system associated with an ingress buffer  $i$  of an OpenFlow switch, (2) expected number of packets queued at an ingress buffer  $i$  of an OpenFlow switch, (3) expected waiting time of a packet in the system of an OpenFlow switch, and (4) expected waiting time of a packet in buffer  $i$  of an OpenFlow switch.

The expected number of packets in the system for buffer  $i$ , denoted by  $\mathcal{L}_{s,i}$ , is expressed as follows:

$$\mathcal{L}_{s,i} = \sum_{m=0}^{K_i} mq_{m,i} = q_i \left( \frac{\lambda_i}{\mu} \right) \left[ \frac{1 - \left[ 1 + K_i \left( 1 - \frac{\lambda_i}{\mu} \right) \right] \left( \frac{\lambda_i}{\mu} \right)^{K_i}}{\left( 1 - \frac{\lambda_i}{\mu} \right) \left[ 1 - \left( \frac{\lambda_i}{\mu} \right)^{K_i+1} \right]} \right] \quad (4.7)$$

The expected number of packets in the buffer  $i$  is denoted by  $\mathcal{L}_{q,i}$ . The expected number of packets in service, which is the number of packets getting matched against the ingress and egress flow-table entries, denoted as  $\mathcal{L}_{pr,i}$ . We calculate  $\mathcal{L}_{pr,i}$  as the probability that the processing unit of OpenFlow switch is busy, and expressed as —  $\mathcal{L}_{pr,i} = q_i \left( \frac{\lambda_i}{\mu} \right)$ .

Therefore,  $\mathcal{L}_{q,i}$  is expressed as follows:

$$\mathcal{L}_{q,i} = q_i \left( \frac{\lambda_i}{\mu} \right) \left[ \frac{\frac{\lambda_i}{\mu}}{1 - \frac{\lambda_i}{\mu}} - \frac{(K_i+1) \left( \frac{\lambda_i}{\mu} \right)^{K_i}}{1 - \left( \frac{\lambda_i}{\mu} \right)^{K_i+1}} \right] \quad (4.8)$$

Based on Equation (4.8), the maximum buffer size and the maximum packet traffic intensity of buffer  $i$  of an OpenFlow switch are evaluated in Theorems 4.2 and 4.3, respectively.

**Theorem 4.2.** *For a fixed packet traffic intensity,  $\frac{\lambda_i}{\mu}$ , of buffer  $i$ , the maximum buffer size,  $K_i$ , needs to satisfy the following constraint:*

$$(K_i + 1) \ln \left( \frac{\lambda_i}{\mu} \right) + 1 = \left( \frac{\lambda_i}{\mu} \right)^{K_i+1} \quad (4.9)$$

---

#### 4. Buffer Size Analysis of SDN Switches

*Proof.* Considering that traffic intensity is fixed, we need to evaluate the maximum buffer size  $\mathcal{L}_{q,i}^{max}$  required for minimizing the packet drop rate. Mathematically,

$$\mathcal{L}_{q,i}^{max} = \max_{K_i} \mathcal{L}_{q,i} \quad (4.10)$$

Hence, we take first order derivative of  $\mathcal{L}_{q,i}$  with respect to  $K_i$  and put  $\frac{d\mathcal{L}_{q,i}}{dK_i} = 0$ . Thereafter, considering that  $\frac{\lambda_i}{\mu} \neq 1$  and  $q_i \neq 0$ , we get Equation (4.9).

Additionally, performing second order derivative of  $\mathcal{L}_{q,i}$  with respect to  $K_i$ , we get that  $(1 - (\frac{\lambda_i}{\mu})^{K_i+1}) > 0$  and  $\frac{d^2\mathcal{L}_{q,i}}{dK_i^2} < 0$ . Hence, we proof that for a fixed packet arrival intensity, the maximum value of  $K_i$  holds the constraint mentioned in Equation (4.9).  $\square$

**Theorem 4.3.** *For a fixed buffer size,  $K_i$ , of buffer  $i$ , the maximum packet traffic intensity,  $\frac{\lambda_i}{\mu}$ , needs to satisfy the following constraint:*

$$\left[ \frac{1 - (\frac{\lambda_i}{\mu})^{k_i+1}}{1 - (\frac{\lambda_i}{\mu})} \right]^2 = (k_i + 1)^2 \left[ \frac{(\frac{\lambda_i}{\mu})^{k_i-1}}{2 - (\frac{\lambda_i}{\mu})} \right] \quad (4.11)$$

*Proof.* Considering that buffer size is fixed, our objective is:

$$\max \mathcal{L}_{q,i} \quad (4.12)$$

while satisfying the constraint that  $\frac{\lambda_i}{\mu} < 1$ . Hence, taking first order derivative of  $\mathcal{L}_{q,i}$  with respect to  $\frac{\lambda_i}{\mu}$  and considering  $\frac{d\mathcal{L}_{q,i}}{d(\frac{\lambda_i}{\mu})} = 0$ ,  $\frac{\lambda_i}{\mu} \neq 0$ , and  $q_i \neq 0$ , we get the condition for optimum value of  $\frac{\lambda_i}{\mu}$  as mentioned in Equation (4.11).

Additionally, performing second order derivative of  $\mathcal{L}_{q,i}$  with respect to  $K_i$ , we get that the maximum packet traffic intensity,  $\frac{\lambda_i}{\mu}$ , follows the constraint given in Equation (4.11), while taking into consideration that the following inequality holds:

$$(K_i + 1)^2 \sqrt{\frac{(\frac{\lambda_i}{\mu})^{k_i-1}}{2 - \frac{\lambda_i}{\mu}}} \leq K_i + (K_i + 2) \left(\frac{\lambda_i}{\mu}\right)^{k_i+1} \quad (4.13)$$

### 4.3. Case Study

---

□

We define the waiting time of a packet in an OpenFlow switch as the time unit spent by a packet in an OpenFlow switch before leaving the output port. Therefore, the expected waiting time of a packet directed to buffer  $i$  in an OpenFlow switch,  $\mathcal{W}_{s,i}$ , is defined as  $\mathcal{W}_{s,i} = \frac{\mathcal{L}_{s,i}}{\lambda_i}$ .

We define the waiting time at buffer  $i$  as the time unit spent by a packet in an OpenFlow switch before entering into the ingress flow-table. The expected waiting time of a packet at buffer  $i$  of an OpenFlow switch,  $\mathcal{W}_{q,i}$ , is defined as  $\mathcal{W}_{q,i} = \frac{\mathcal{L}_{q,i}}{\lambda_i}$ .

### 4.3 Case Study

We considered two cases — (1) single ingress port, i.e.,  $I = 1$ , and (2)  $I$  ingress ports with equal packet traffic intensity, where  $I \geq 2$ . These cases are discussed briefly in the following section.

#### 4.3.1 Case I : $I = 1$

We consider that in an OpenFlow switch, there is a single ingress port. In other words, the number of ingress buffer is one, i.e.,  $I = 1$ . Hence, the expected number of packets in the system,  $\mathcal{L}_{s,1}$ , and in the buffer,  $\mathcal{L}_{q,1}$ , are as follows:

$$\mathcal{L}_{s,1} = \left(\frac{\lambda}{\mu}\right) \left[ \frac{1 - [1 + K(1 - \frac{\lambda}{\mu})] (\frac{\lambda}{\mu})^K}{(1 - \frac{\lambda}{\mu}) [1 - (\frac{\lambda}{\mu})^{K+1}]} \right] \quad (4.14)$$

$$\mathcal{L}_{q,1} = \left(\frac{\lambda}{\mu}\right) \left[ \frac{\frac{\lambda}{\mu}}{1 - \frac{\lambda}{\mu}} - \frac{(K+1)(\frac{\lambda}{\mu})^K}{1 - (\frac{\lambda}{\mu})^{K+1}} \right] \quad (4.15)$$

where  $\lambda$  and  $\mu$  are the average packet arrival rate and service rate, respectively, of an OpenFlow switch, and  $K$  defines the buffer size of the ingress port. Additionally, the expected waiting time of a packet in an OpenFlow switch before reaching the output

port,  $\mathcal{W}_{s,1}$ , and the waiting time at ingress buffer of an OpenFlow switch,  $\mathcal{W}_{q,1}$ , are defined as follows:

$$\mathcal{W}_{s,1} = \frac{\mathcal{L}_{s,1}}{\lambda} \quad \text{and} \quad \mathcal{W}_{q,1} = \frac{\mathcal{L}_{q,1}}{\lambda} \quad (4.16)$$

### 4.3.2 Case II : $I \geq 2$

We consider that there are at least two ingress ports ( $I \geq 2$ ) in an OpenFlow buffer. Additionally, we consider that each arrived packet has an equal probability of being at any of the available ingress port or buffer. Therefore, the probability of a packet being queued at buffer  $i$ ,  $q_i$ , is  $\frac{1}{I}$ .

$$\mathcal{L}_{s,i} = \frac{\lambda}{I^2\mu} \left[ \frac{1 - [1 + K(1 - \frac{\lambda}{I\mu})] (\frac{\lambda}{I\mu})^K}{(1 - \frac{\lambda}{I\mu}) [1 - (\frac{\lambda}{I\mu})^{K+1}]} \right] \quad (4.17)$$

$$\mathcal{L}_{q,i} = \frac{\lambda}{I^2\mu} \left[ \frac{\frac{\lambda}{I\mu}}{1 - \frac{\lambda}{I\mu}} - \frac{(K+1)(\frac{\lambda}{I\mu})^K}{1 - (\frac{\lambda}{I\mu})^{K+1}} \right] \quad (4.18)$$

where  $\lambda_1 = \dots = \lambda_I = \frac{\lambda}{I}$  and  $K_1 = \dots = K_I = K$ . Hence,  $\lambda$  and  $K$  are constants for a specific OpenFlow switch. Additionally, the expected waiting time of a packet in an OpenFlow switch before reaching output port,  $\mathcal{W}_{s,i}$ , and the waiting time at ingress buffer  $i$  of an OpenFlow switch,  $\mathcal{W}_{q,i}$ , are defined as follows:

$$\mathcal{W}_{s,i} = \mathcal{L}_{s,i} \frac{I}{\lambda} \quad \text{and} \quad \mathcal{W}_{q,i} = \mathcal{L}_{q,i} \frac{I}{\lambda} \quad (4.19)$$

## 4.4 Performance Evaluation

In this section, we analyze the required buffer size of an OpenFlow switch-based system with varying packet arrival and processing rate in SD-DCN. We evaluate the performance of OPUS scheme for an OpenFlow switch, based on the parameters such as maximum arrival rate, minimum buffer size, and maximum waiting time. Generic test-bed infor-

#### 4.4. Performance Evaluation

---

mation for OPUS is provided in Table 4.1. For simplicity, we evaluate the buffer size requirement of a single OpenFlow switch in SD-DCN. Additionally, we consider that each packet gets queued at buffer  $i \in [1, I]$  of an OpenFlow switch with a probability  $q_i$ . We consider that each packet selects a buffer, i.e., queue,  $i$ , randomly.

**Table 4.1:** System Specification

Parameter	Value
Processor	Intel(R) Core(TM) i5-2500 CPU @ 3.30 GHz
RAM	4 GB DDR3
Disk Space	500 GB
Operating System	Ubuntu 16.04 LTS
Application Software	MATLAB 2015b

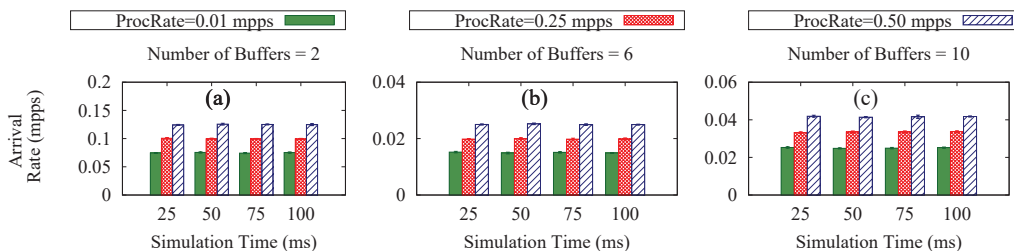
**Table 4.2:** Simulation Parameters

Parameter	Value
Number of OpenFlow switch	1
Number of buffers	2, 6, 10
Total buffer size	0.5, 0.75, 1 million packets
Packet arrival rate	0.15, 0.20, 0.25 <i>mpps</i>
Packet processing rate	0.01, 0.025, 0.05 <i>mpps</i>
Packet size	1500 <i>Byte</i> [9]
Simulation Duration	100 <i>ms</i>

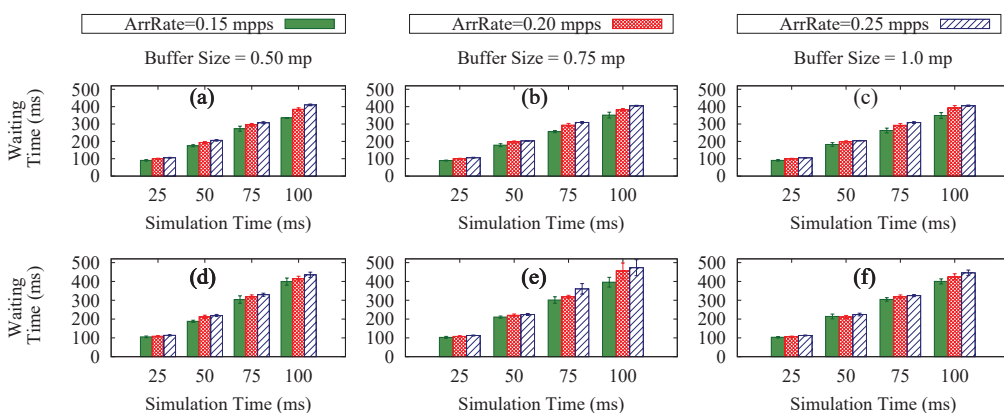
##### 4.4.1 Simulation Parameters

We simulated the performance of an OpenFlow switch-based system in SD-DCN, where each OpenFlow switch has multiple numbers of queues such as 2 and 6, as mentioned in Table 4.2. The total size of the buffer for each OpenFlow switch is varied in 0.5–1 million packets (*mp*). On the other hand, the packet processing rate is varied in 0.01 – 0.05 million packets per second (*mpps*), as mentioned in Table 4.2. The size of each packet is considered as 1500 *bytes* [9]. We simulate OPUS for different simulation durations — 25, 50, 75, 100 *ms*.

## 4. Buffer Size Analysis of SDN Switches



**Figure 4.3:** Maximum Arrival Rate per OpenFlow Switch.



**Figure 4.4:** Maximum Waiting Time per OpenFlow Switch: In (a), (b) and (c), and in (d), (e) and (f), number of buffers are 2 and 6, respectively; The buffer size per queue are 0.50, 0.75, and 1.0 mp in (a) and (d), (b) and (e), and (c) and (f), respectively.

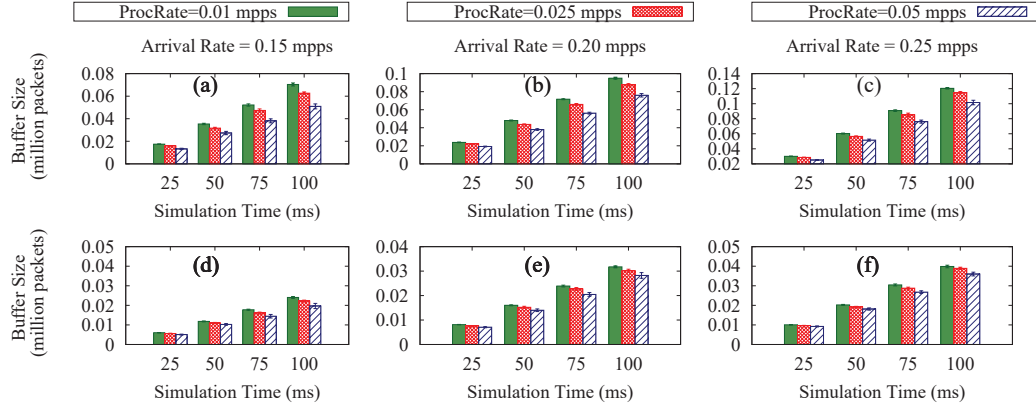
### 4.4.2 Performance Metrics

We evaluated the performance of an OpenFlow switch with the different number of queues or buffers such as 2, 6, 10, in the proposed I-M/M/1/K queue-based scheme, OPUS, while considering the following parameters:

- **Maximum Arrival Rate (ArrRate):** The maximum arrival rate depends on the average buffer size and the maximum processing rate. It varies proportionally with the average buffer size of an OpenFlow switch. Additionally, the maximum arrival rate varies proportionally with the number of buffers and the processing rate of an OpenFlow switch.



#### 4.4. Performance Evaluation



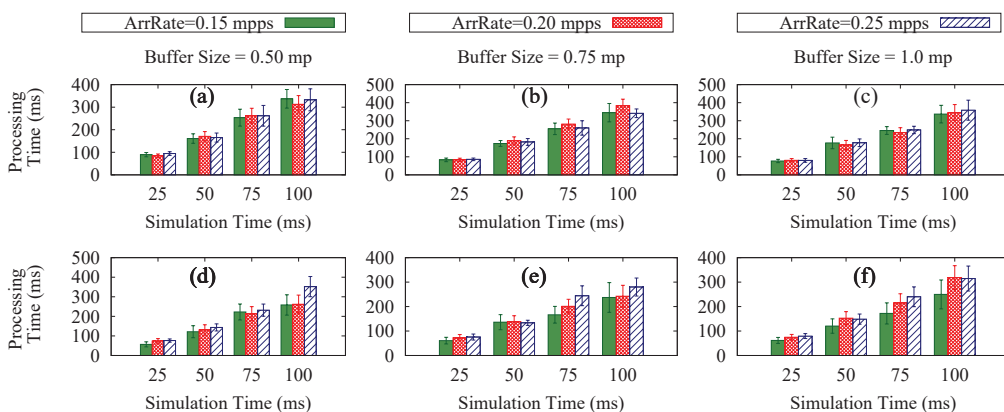
**Figure 4.5:** Minimum Buffer Size per OpenFlow Ingress Port: In (a), (b) and (c), and in (d), (e) and (f), number of buffers are 2 and 6, respectively; The packet arrival rates are 0.15, 0.20, and 0.25 mpps in (a) and (d), (b) and (e), and (c) and (f), respectively.

- Minimum Buffer Size (BuffSize):** The OpenFlow switches have Ternary Content-Addressable Memory (TCAM) memory, which is costly. Hence, we need to evaluate the minimum buffer size requirement of an OpenFlow switch for an optimum traffic intensity.
- Maximum Waiting Time:** The performance of an OpenFlow switch mostly depends on the waiting time of a packet in the system. With the increase in the waiting time, the performance of an OpenFlow switch degrades. Hence, we need to evaluate the maximum waiting time of a packet in an OpenFlow switch.

#### 4.4.3 Results and Discussions

For simulation, we considered that the packets enter through the ingress port of an OpenFlow switch, and get forwarded *randomly* to any of the available buffers, before getting matched against the ingress flow-tables. Thereafter, based on the table-hit and table-miss entry, the packets are processed. Additionally, we consider that the packets, which are forwarded to the SDN controller, get queued in the ingress buffers as newly arrived packets.

#### 4. Buffer Size Analysis of SDN Switches



**Figure 4.6:** Maximum Processing Time per OpenFlow Switch: In (a), (b) and (c), and in (d), (e) and (f), number of buffers are 2 and 6, respectively; The buffer size per queue are 0.50, 0.75, and 1.0 mp in (a) and (d), (b) and (e), and (c) and (f), respectively.

From Figure 4.3, we observe that the maximum arrival rate, which can be handled by an OpenFlow switch, increases by 26.15-30.4% with the increase in the processing rate of an OpenFlow switch by *two times*. Additionally, we observe that with the increase in the number of buffers, the maximum packet arrival rate per buffer decreases. However, the maximum packet arrival rate in an OpenFlow switch remains the same, while considering that the buffer size of an OpenFlow switch is constant. Therefore, we conclude that the arrival rate of an OpenFlow switch remains constant with a fixed buffer size and fixed processing rate.

From Figure 4.5, we observe that with the increase in the arrival rate, the minimum buffer size requirement increases. On the other hand, for a fixed arrival rate, with the increase in processing rate, the buffer size requirement increases up to the processing rate 0.03-0.35 *mpps*. Thereafter, the minimum buffer size remains constant. Hence, we conclude that the optimum arrival rate and processing rate of an OpenFlow switch lie in the ranges 0.20-0.25 *mpps* and 0.03-0.35 *mpps*, respectively. The minimum buffer size required is in the range 0.67-0.80 million packets.

Figure 4.6 shows that for buffer size 0.75 million packets, the processing rate of an OpenFlow switch is less while considering that the packet arrival rate varies in the range

#### 4.5. Concluding Remarks

---

0.15-0.25 *mpps*. On the other hand, from Figure 4.6, we infer that the packet processing rate of an OpenFlow switch varies insignificantly. Hence, we conclude that for the packet arrival rate in an OpenFlow switch with the rate of 0.15-0.25 *mpps*, the optimum number of buffers is *two*. Additionally, the optimum buffer size of an OpenFlow switch is 0.75 million packets, i.e., 1.125 *GB*, while considering that each packet is of size 1500 *bytes*, as mentioned in Table 4.2. In Figure 4.4, we observe that the packet waiting time is less for an OpenFlow switch with buffer size 0.50 million packets, as few packets get dropped due to insufficient buffer space at an OpenFlow switch. On the other hand, in Figure 4.4, we observe that the waiting time of an OpenFlow is similar for OpenFlow switches with buffer size 0.75 and 1.00 million packets. Hence, we conclude that the minimum waiting time at an OpenFlow switch can be ensured with a buffer size of 0.75 million packets, i.e., 1.125 *GB*. These analytical results confirm with the OpenFlow specification given in Refs [9] and [87].

We maintain that the performance of an OpenFlow switch can be improved with the packet arrival and processing rate of 0.20-0.25 *mpps* and 0.30-0.35 *mpps*, respectively. On the other hand, the optimum buffer size of an OpenFlow switch is 0.75 million packets, i.e., 1.125 *GB*, as observed using OPUS.

#### 4.5 Concluding Remarks

In this Chapter, we analyzed the optimum buffer size of an OpenFlow switch in order to ensure quality-of-service in OpenFlow systems. We analyzed the optimum packet arrival and processing rates and the average waiting of packets in an OpenFlow switch-based system. In OPUS, we modeled the architecture of an OpenFlow switch as a I-M/M/1/K queue, while considering that there are  $I$  ingress buffers. Each buffer has  $K$  memory blocks in an OpenFlow switch. We analyzed the optimum number of buffers with the optimum value of each buffer. Additionally, we evaluated the optimum packet arrival and processing rates of an OpenFlow switch using OPUS. Simulation-based analysis

#### 4. Buffer Size Analysis of SDN Switches

---

exhibited that with two times increase in packet processing rate, the packet arrival rate can be increased by 26.15-30.4%. We inferred that for an OpenFlow system, the minimum buffer size is 0.75 million packets with the maximum packet arrival and the minimum processing rate of 0.20-0.25 million packets per second (mpps) and 0.30-0.35 mpps, respectively, and the maximum packet waiting time is 0.173-0.249 second.

s

## Chapter 5

# QoS-Aware Data Traffic Management

In this Chapter, we present two data traffic management schemes — network-specific QoS-aware data traffic management (TROD) and flow-specific QoS-aware data traffic management (FlowMan) which consider to optimize the flow-specific and network-specific throughput and delay in SD-DCN. Both of these schemes are suitable for SD-DCN in the presence of heterogeneous mice and elephant flows generated by IoT devices.

This Chapter is organized as follows. Initially, we model the proposed scheme, TROD, in Section 5.1. The design of TROD is presented in Section 5.1.1. Section 5.1.3 discusses formulation of TROD using evolutionary game [90] along with the designed algorithm for TROD in Section 5.1.5. We evaluate the performance of TROD to the benchmark schemes in Section 5.1.6. Thereafter, we model the proposed scheme, FlowMan, in Section 5.2. We present the design of FlowMan in Section 5.2.1. In Section 5.2.3, we present the generalized Nash bargaining game [91]-based game-theoretic model formulation of FlowMan. The performance evaluation of FlowMan is discussed in Section 5.2.8. Finally, Section 5.3 concludes this Chapter.

## 5.1 TROD: The Throughput-Optimal Data Traffic Management Scheme

In this work, we introduce a game theory-based dynamic data traffic management scheme, *named TROD*, for minimizing network delay and maximizing network throughput in SD-DCN in the presence of IoT devices. We use an *evolutionary game-theoretic* [90] approach to deciding the optimal data traffic volume which needs to be handled by the switches, while considering that the data generation rate for each IoT device is known *a priori*. Moreover, the evolutionary game helps to develop the intermediate sub-optimal problem for efficient data traffic management. Thereafter, with the help of linear programming, we evaluate the *optimal time distribution matrix*, which enables the optimal data traffic distribution in SD-DCN in the presence of IoT-devices.

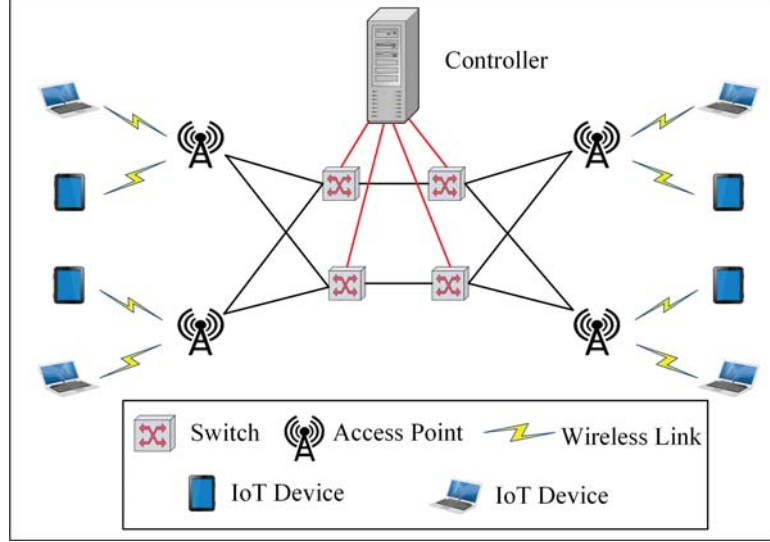
### 5.1.1 System Model

We consider an SD-DCN having a single controller and multiple SDN switches. The schematic diagram of SD-DCN architecture is shown in Figure 5.1. We consider that the IoT devices are connected with the SDN switches through the access point (APs). These IoT devices are heterogeneous in terms of data generation rate. We consider a stochastic process, where the IoT devices are static for a fixed time duration. Each AP is connected with *multiple* SDN switches. The data traffic generated by the IoT devices are forwarded to one of the connected SDN switches via available AP. Thereafter, the data traffic is processed by the SDN switches according to the flow-rule entries and forwarded to the backhaul network for further processing.

In the case of table-miss, meta-data of the data traffic is forwarded to the controller, and it installs the corresponding rule to one of the switches. Additionally, due to limited TCAM, there is a limitation of the maximum number of rules that can be installed in an SDN switch. For example, according to the OpenFlow specification version 1.5.0 [5],

## 5.1. TROD: The Throughput-Optimal Data Traffic Management Scheme

maximum of 8000 rules can be installed in an OpenFlow switch, while considering that 45 match fields are present in each rule in the flow table.



**Figure 5.1:** Schematic Diagram of SD-DCN in the Presence of IoT-Devices

As shown in Figure 5.1, with the change in the position of edge devices, i.e., IoT devices, the installed rules need to be changed dynamically by the controller. In such a situation, the throughput of the SDN switch gets affected depending on the number of active flow-rules in the flow table and the volume of the data traffic. We define *active flow-rule* as mentioned in Definition 5.1.

**Definition 5.1.** We define a flow-rule to be active at time instant  $t$  if that rule is used for forwarding data traffic within a time duration of  $[t - \delta, t]$ , where  $\delta > 0$ .

We consider that a sub-tree of IoT-enabled SD-DCN comprises of a single controller, multiple SDN switches, and multiple IoT devices. Each switch  $s \in \mathcal{S}$ , where  $\mathcal{S}$  is the set of available switches, is capable of installing maximum  $R_s^{max}$  number of rules. Therefore, the maximum number of rules  $\mathbb{R}^{max}$ , which can be accommodated is expressed as follows:

$$\mathbb{R}^{max} = \sum_{s \in \mathcal{S}} R_s^{max} \quad (5.1)$$

---

## 5. QoS-Aware Data Traffic Management

Moreover, we consider that each IoT device  $n \in \mathcal{A}$ , where  $\mathcal{A}$  is the set of available IoT devices in SD-DCN, generates  $f_n$  number of data traffic flows. The data generation-rate of each data traffic flow  $i$ , where  $0 \leq i \leq f_n$  and  $n \in \mathcal{A}$ , is denoted as  $\lambda_i$ . Additionally, we consider that the volume of data associated with each data traffic flow  $i$ , where  $0 \leq i \leq f_n$  and  $n \in \mathcal{A}$ , is denoted by  $\nu_i$ . Therefore, we have:

$$\nu_i = \sum_{t=1}^T \lambda_i \quad (5.2)$$

We consider that each data traffic flow  $i$  generated by the IoT devices is *mutually exclusive*. Therefore, the data traffic management by the controller needs to ensure the following constraint:

$$\sum_{n \in \mathcal{A}} f_n \leq \mathbb{R}^{max} \quad (5.3)$$

We consider that each switch  $s \in \mathcal{S}$  handles  $F_s$  number of flow-rules. Additionally, the incoming-flow rate at time instant  $t$  for each switch  $s$  is denoted as  $\lambda_s(t)$ . Hence, the incoming volume of data traffic  $V_s$  handled by each switch  $s$  is defined as follows:

$$V_s = \sum_{j=1}^{F_s} \nu_j = \sum_{t=1}^T \lambda_s(t) \quad (5.4)$$

where  $T$  is the time duration for which the IoT devices remain static. On the other hand, each switch  $s$  has a processing limit of  $\mu_s$  amount of data per unit time. Therefore, in order to ensure optimal throughput of the network, the controller needs to ensure that the data traffic is properly distributed among the available switches.

### 5.1.2 Justification for Using Evolutionary Game

For efficient data traffic management, we need to ensure that the load of the network, i.e., the volumetric data, is optimally distributed among the available switches. Hence, in order to reduce the processing delay in the switch, the controller tries to reduce the



## 5.1. TROD: The Throughput-Optimal Data Traffic Management Scheme

---

load on each switch. However, the controller also needs to ensure the overall throughput of the network. Thereby, our objective is as follows:

$$\max_{\sum \mu_s} \min_{\lambda_s} \sum_{s \in \mathcal{S}} V_s \quad (5.5)$$

while satisfying the constraints given in Equations (5.3) and (5.4). Additionally, we need to satisfy the following constraints:

$$\lambda_s \geq 0 \text{ and } \mu_s \leq \mu^{max} \quad (5.6)$$

$$\sum_{s \in \mathcal{S}} x_{s,i}(t) = 1 \quad \text{and} \quad F_s = \sum_{n \in \mathcal{A}} \sum_{i=1}^{f_n} x_{s,i} \quad (5.7)$$

where  $\mu^{max}$  is the maximum processing rate of the SDN switches; and  $x_{s,i}(t)$  denotes the association vector of flow-rule  $F_s$ , and is a binary variable. We define:

$$x_{s,i}(t) = \begin{cases} 1, & \text{if Rule } i \text{ is installed at switch } s \\ 0, & \text{otherwise} \end{cases} \quad (5.8)$$

Therefore, we infer that the aforementioned problem is an *integer programming* problem, in which the variable  $x_{s,i}(t)$  can take only binary values —  $\{0, 1\}$ . Hence, this problem is an *NP-complete* problem [92]. Thereby, we use an evolutionary game-theoretic approach in order to reduce it to a *linear programming* problem, and achieve a sub-optimal solution, i.e., ensuring optimal throughput of the network, in polynomial time.

### 5.1.3 Game Formulation

In order to achieve a sub-optimal solution, we use an *evolutionary game theoretic* approach. In the proposed dynamic data traffic management scheme, named TROD, the IoT devices act as the players and choose the set of optimal SDN switches, i.e., strategies in TROD, for forwarding data with the help of SDN controller. Here, the controller acts

as a centralized coordinator. In TROD, we consider that the population share, i.e., the total volume of data traffic, needs to be divided among the available switches. Therefore, the population share  $y_s(\omega)$  of each switch  $s \in \mathcal{S}$  is defined as follows:

$$y_s(\omega) = \frac{V_s(\omega)}{\sum_{s \in \mathcal{S}} V_s} \quad (5.9)$$

where  $\omega$  is the evolutionary iteration.

### 5.1.3.1 Utility Function of Each Switch

Utility value  $\phi_s(\omega)$  signifies the fitness function for switch  $s$ . We consider that  $\phi_s(\omega)$  varies linearly with the population share of switch  $s$ . On the other hand, with the increase in number of active flow-rules  $F_s(\omega)$  installed, the utility value decreases. For example, switches  $s_1$  and  $s_2$  have population share of  $y_{s_1}(\omega)$  and  $y_{s_2}(\omega)$ , respectively. Additionally, the number of rules installed in the switches  $s_1$  and  $s_2$  are  $F_{s_1}(\omega)$  and  $F_{s_2}(\omega)$ , respectively. Hence, considering that  $F_{s_1}(\omega) = F_{s_2}(\omega)$ , we get  $\phi_{s_1}(\omega) \succeq \phi_{s_2}(\omega)$ , where  $y_{s_1}(\omega) \geq y_{s_2}(\omega)$ . On the other hand, we observe  $\phi_{s_1}(\omega) \preceq \phi_{s_2}(\omega)$ , while considering that  $F_{s_1}(\omega) \geq F_{s_2}(\omega)$  and  $y_{s_1}(\omega) = y_{s_2}(\omega)$ . Hence, we define the utility function as follows:

$$\phi_s(\omega) = y_s(\omega) \left( 1 - \frac{F_s(\omega)}{R_s^{max}} \right) \quad (5.10)$$

Thereafter, the controller calculates average payoff  $\bar{\phi}(\omega)$  of the population using the following equation:

$$\bar{\phi}(\omega) = \sum_{s \in \mathcal{S}} y_s(\omega) \phi_s(\omega) \quad (5.11)$$

### 5.1.3.2 Replicator Dynamics of TROD

In TROD, each switch acts as a replicator, and tries to mutate and evolve over successive iteration and reach the evolutionary equilibrium. Dynamic of the change in population

## 5.1. TROD: The Throughput-Optimal Data Traffic Management Scheme

---

share is obtained using *replicator dynamics*. In TROD, the replicator dynamics of each switch  $s$  is represented as follows:

$$\dot{y}_s(\omega) = \sigma y_s(\omega) (\phi_s(\omega) - \bar{\phi}(\omega)) \quad (5.12)$$

where  $\sigma$  is the evolution controlling factor. In TROD,  $\sigma$  value determines the permissible rate of change in population share in two successive evolutionary iterations.

### 5.1.3.3 Reduced Sub-Optimal Problem

By using evolutionary game theoretic approach, we get the population share  $y_s^*$  of each switch  $s$  at equilibrium, while satisfying the following constraint:

$$\dot{y}_s(\omega)|_{y_s(\omega)=y_s^*} \approx 0 \quad (5.13)$$

Hence, the vector of the volume of data traffic, which needs to be handled by the switches, is defined as follows:

$$\mathbf{V} = \mathbf{y} \sum_{s \in \mathcal{S}} V_s \quad (5.14)$$

where  $\mathbf{V} = [V_1^* \cdots V_{|\mathcal{S}|}^*]^T$  and  $\mathbf{y} = [y_1^* \cdots y_{|\mathcal{S}|}^*]^T$ .

Therefore, we yield a system of linear equations, which is easily solvable using linear programming. The system of linear equations is expressed as follows:

$$\boldsymbol{\lambda} \mathbb{T}^T = \mathbf{V} \quad (5.15)$$

where  $\boldsymbol{\lambda} = [\lambda_1 \cdots \lambda_{(\sum_{n \in \mathcal{A}} f_n)}]$ , and  $\mathbb{T}^T$  defines the transpose of matrix  $\mathbb{T}$ . We define time-distribution matrix  $\mathbb{T}$  as follows:

$$\mathbb{T} = \begin{bmatrix} \Delta t_{11} & \cdots & \Delta t_{(\sum f_n)1} \\ \vdots & \ddots & \vdots \\ \Delta t_{1|\mathcal{S}|} & \cdots & \Delta t_{(\sum f_n)|\mathcal{S}|} \end{bmatrix} \quad (5.16)$$

where  $\Delta t_{is}$  defines the time duration for which data traffic flow  $i$  gets forwarded by switch  $s$ . Additionally, Equation (5.15) needs to satisfy the following constraint:

$$\sum_{n \in \mathcal{A}} \sum_{i=1}^{f_n} \Delta t_{is} = T, \quad \forall s \in \mathcal{S} \quad (5.17)$$

#### 5.1.4 Theoretical Analysis:

In this section, we analyze TROD, theoretically, while evaluating the existence of evolutionary equilibrium. As mentioned earlier, at evolutionary equilibrium, the change in population share  $y_s(\omega)$  of each switch  $s$  becomes zero. Therefore, we get:

$$\sigma y_s(\omega) \left( \phi_s(\omega) - \bar{\phi}(\omega) \right) = 0 \quad (5.18)$$

We consider that  $\sigma > 0$  and is a constant, and each switch  $s$  has a positive population share, i.e.,  $y_s(\omega) > 0$ . Therefore, Equation (5.18) can be written as follows:

$$\phi_s(\omega) - \bar{\phi}(\omega) = 0 \quad (5.19)$$

By solving Equation(5.19), we get:

$$(y_s^*)^2 - y_s^* + \frac{\sum_{s' \in \mathcal{S}/\{s\}} (y_{s'}^*)^2 \left[ 1 - \frac{F_{s'}}{R_s^{max}} \right]}{1 - \frac{F_s}{R_s^{max}}} = 0 \quad (5.20)$$

Hence, theoretically, we yield that the optimal population share  $y_s^*$  of each switch  $s$  is as follows:

## 5.1. TROD: The Throughput-Optimal Data Traffic Management Scheme

---



---

### Algorithm 5.1 Algorithm for TROD Scheme

---

**INPUTS:**

- 1:  $\mathbb{N}$  ▷ Set of available IoT devices
- 2:  $\mathcal{S}$  ▷ Set of available SDN switches
- 3:  $f_n$  ▷ Number of flows of IoT-device  $n$
- 4:  $F_s$  ▷ Flow-rules installed in switch  $s$
- 5:  $\mathbb{R}^{max}$  ▷ Maximum number of rules in sub-tree of SD-DCN
- 6:  $R_s^{max}$  ▷ Maximum number of rules in switch  $s$
- 7:  $\lambda_i$  ▷ Generation-rate of data traffic flow  $i$
- 8:  $\sigma$  ▷ Evolution controlling factor

**OUTPUT:**

- 1:  $\mathbb{T}$  ▷ Time-distribution matrix

**PROCEDURE:**

- 1:  $\omega \leftarrow 1$
  - 2: Randomly map each data traffic flow  $0 \leq i \leq f_n$ , where  $n \in \mathcal{A}$  to any switch  $s \in \mathcal{S}$
  - 3: **for** Each  $s \in \mathcal{S}$  **do**
  - 4:     Calculate  $V_s(\omega)$  using Equations (5.2) and (5.4)
  - 5:     Calculate population share  $y_s(\omega)$  using Equation (5.9)
  - 6: **end for**
  - 7: **do**
  - 8:     **for** Each  $s \in \mathcal{S}$  **do**
  - 9:         Calculate utility value  $\phi_s(\omega)$  using Equation (5.10)
  - 10:     **end for**
  - 11:     Calculate average payoff  $\bar{\phi}(\omega)$  of the population using Equation (5.11)
  - 12:     **for** Each  $s \in \mathcal{S}$  **do**
  - 13:         Calculate replicator dynamics  $y_s(\omega)$  using Equation (5.12)
  - 14:          $\omega \leftarrow \omega + 1$
  - 15:          $y_s(\omega) \leftarrow y_s(\omega - 1) - y_j(\omega - 1)$
  - 16:     **end for**
  - 17: **while** ( $y_s(\omega) \approx 0$ )
  - 18:      $y_s^* \leftarrow y_s(\omega)$
  - 19:     Calculate  $\mathbf{V}$  using Equation (5.14)
  - 20:     Calculate  $\mathbb{T}$  using Equation (5.15)
  - 21: **return**  $\mathbb{T}$ ;
-

$$y_s^* = \frac{1 \pm \sqrt{1 - 4\zeta}}{2} \quad (5.21)$$

where  $\zeta = \left[ \frac{\sum_{s' \in \mathcal{S}/\{s\}} (y_{s'}^*)^2 \left[ 1 - \frac{F_{s'}}{R_{s'}^{max}} \right]}{1 - \frac{F_s}{R_s^{max}}} \right]$ . Therefore, we get that TROD ensures evolutionary equilibrium. Based on the derived  $\{y_s^* | \forall s \in \mathcal{S}\}$  values, we get  $(|\mathcal{S}| + \sum_{n \in \mathcal{A}} f_n)$  number of linear equations. By solving the linear equations, we can easily obtain the dynamic data traffic distribution over a fixed time duration  $T$ , while ensuring optimal delay at the switch-end and optimal throughput of SD-DCN.

### 5.1.5 Proposed Algorithm:

To achieve the sub-optimal solution of TROD, the controller tries to distribute the data traffic among the available SDN switches. The controller tries to ensure that each switch satisfies the equilibrium condition. The dynamic data traffic management scheme, TROD, ensures delay and throughput-optimal data traffic in IoT-enabled SD-DCN using Algorithm 5.1. In Algorithm 5.1, each data traffic flow  $i$ , where  $1 \leq i \leq f_n$  and  $n \in \mathcal{A}$ , is mapped randomly to one of the available switches. Thereafter, with the help of the controller, each switch decides the optimal population share, i.e., data-traffic volume needed to be handled over the time-period  $T$ . Finally, using the linear equations derived with the help of evolutionary game theory, the controller schedules the data-traffic among the available switches with the help of the time-distribution matrix.

### 5.1.6 Performance Evaluation

In this section, we analyze the performance of TROD with the varying number of data traffic flows and the available switches in SD-DCN. Generic test-bed information for TROD is provided in Table 5.2.

## 5.1. TROD: The Throughput-Optimal Data Traffic Management Scheme

---

### 5.1.6.1 Simulation Parameters

For simulation, we varied the number of SDN switches and the number of IoT-devices as mentioned in Table 5.1. We simulated TROD in the MATLAB simulation platform, as mentioned in Table 5.2. We considered that each flow generates data traffic at the rate of 0.2 million packets per second (*mpps*). We simulated TROD for 100 simulation seconds.

**Table 5.1:** Simulation Parameters

Parameter	Value
Number of SDN switches	4, 8, 12
Number of IoT-devices	50, 100, 150, 200
Number of flow per device	10
Data traffic generation rate	0.2 <i>mpps/flow</i>
Maximum number of flow rules per switch	8000 [5]
Simulation duration	100 <i>sec</i>

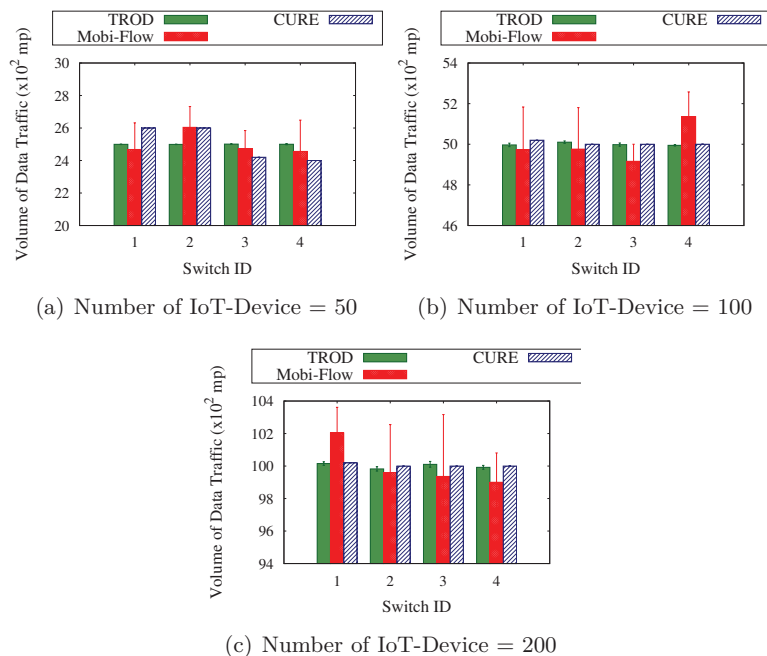
**Table 5.2:** System Specification

Parameter	Value
Processor	Intel(R) Core(TM) i5-2500 CPU @ 3.30 GHz
RAM	4 GB DDR3
Disk Space	500 GB
Operating System	Ubuntu 16.04 LTS
Application Software	MATLAB 2015b

### 5.1.6.2 Benchmarks

The performance of TROD is evaluated by comparing with existing schemes — Mobi-Flow [48] and CURE [93]. In Mobi-Flow, Bera *et al.* [48] proposed a data traffic management scheme in the presence of IoT-devices while predicting the location of the IoT-devices. The authors also studied the dynamic data traffic management, while installing the flow-rules in the SDN switches, dynamically. On the other hand, In CURE, Maity *et al.* [93] proposed a data traffic management scheme while considering the update of

## 5. QoS-Aware Data Traffic Management



**Figure 5.2:** Volume of Data Traffic Processed by Switches with Varied Number of IoT Devices

flow-rules in the switches according to the priorities of the switches.

### 5.1.6.3 Performance Metrics

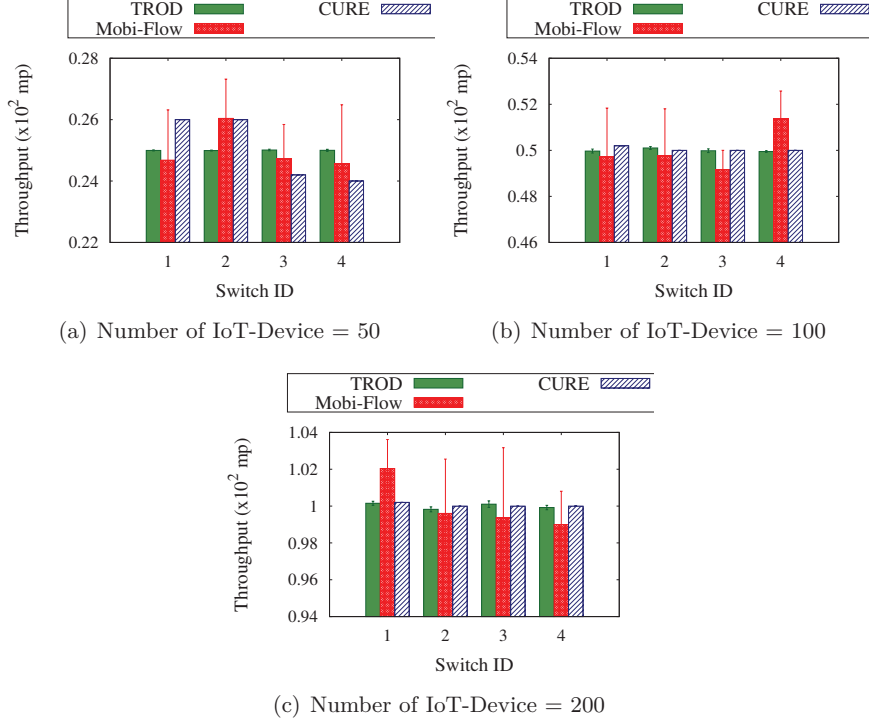
We evaluated the performance of TROD using the following metrics:

**Volume of data traffic processed by each switch:** We consider that with the increase in the volume of data processed by each switch, the queuing delay increases. Hence, we consider the volume of data traffic processed by each switch.

**Throughput of each switch:** The overall throughput of the network depends linearly on the throughput of the individual switch. On the other hand, throughput per switch also signifies the balance factor of the switches in data traffic management.



## 5.1. TROD: The Throughput-Optimal Data Traffic Management Scheme



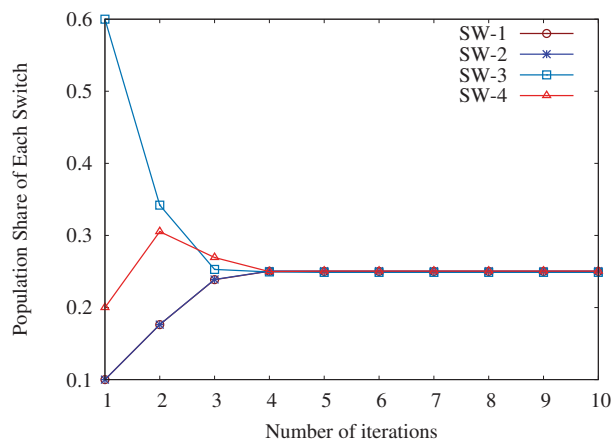
**Figure 5.3:** Throughput of Switches with Varied Number of IoT Devices

### 5.1.6.4 Results and Discussions

For simulations, we considered that the controller updates the flow-tables of each switch, periodically in every 10 simulation seconds.

From Figure 5.2, we observe that with the increase in the number of IoT-devices, the volume of data traffic increases. However, using TROD, the volume of data traffic gets distributed equally among the available switches than using Mobi-Flow and CURE. We observe that TROD distributes the incoming traffic among the available switches equally. Hence, the queuing delay of the network reduces by 23.4-29.7%.

On the other hand, from Figure 5.3, we observe that the throughput of each switch is the same using TROD. Hence, we get that the data traffic in the network is distributed properly using TROD, than using Mobi-Flow and CURE. From Figure 5.3, we observe that the throughput of each switch increases by 23.1-28.78% using TROD than using



**Figure 5.4:** Population Share of each Switch

Mobi-Flow and CURE. Additionally, from Figure 5.4, we observe that the change in the population share of each switch varies linearly with the difference between the actual payoff value of the switch and the average payoff value of the population. Moreover, we observe that within 4-6 iterations, TROD reaches its evolutionary equilibrium.

Therefore, we observe that TROD enhances the performance of network holistically, i.e., minimizes the network delay and maximizes the network throughput, while distributing the incoming data traffic, efficiently and dynamically, in SD-DCN in the presence of IoT-devices.

## 5.2 FlowMan: The QoS-Aware Data Traffic Management Scheme

We consider an SD-DCN with IoT devices and switches connected using access points (APs). These heterogeneous IoT devices are capable of running different applications such as Internet browsing, Email, VoIP calls, and video streaming [94]. As per Federal Communications Commission (FCC) [94], video streaming applications in IoT devices consume almost  $10^4$  times more bandwidth than normal applications such as browsing

## 5.2. FlowMan: The QoS-Aware Data Traffic Management Scheme

and calls. Therefore, we get that the flows generated from these video streaming applications are the elephant flows, whereas the other flows are mice flows. In such a scenario, due to the unbalanced nature of data traffic, the switches handling the elephant flows incur high delay, whereas the switches handling the mice flows incur low throughput. As shown in Figure 5.5, the incoming flows can be discarded due to overflow or can replace the existing flows, which results in high delay or low throughput, respectively. This, in turn, degrades the overall performance of the SD-DCN.

Therefore, in this work, we introduce a QoS-aware stochastic data flow management scheme, named FlowMan, for SD-DCN in the presence of heterogeneous flows. We use a *generalized Nash bargaining game* to decide the Pareto optimal data rate to be allocated to each SDN switch while considering the heterogeneous flows within the one-hop network. Here, using a generalized Nash bargaining game, we achieve an intermediate Pareto optimal throughput for the switches. Thereafter, using a distributed heuristic method, i.e., a method for solving the bounded Knapsack problem, we decide the switch and flow-rule association for ensuring optimal data flow management in SD-DCN.

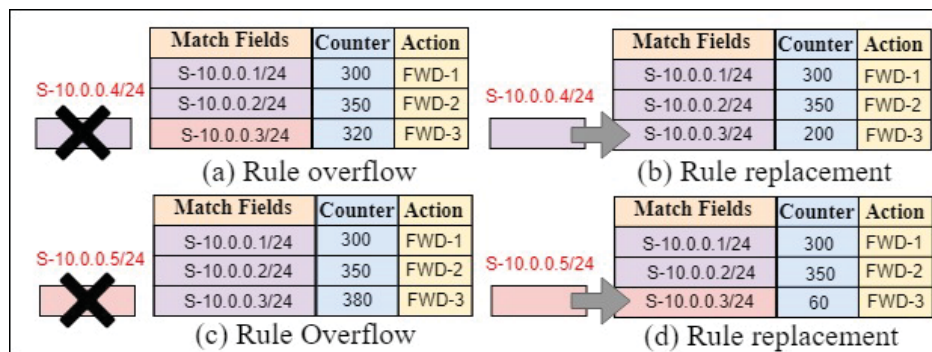


Figure 5.5: Issues in Heterogeneous Flow Management in SD-DCN

### 5.2.1 System Model

We consider a part of SD-DCN with multiple switches, a centralized controller, and multiple IoT devices and the SDN switches connected using access points (APs). We

consider that these IoT devices generate heterogeneous data-traffic such as Internet browsing, Email, VoIP calls, and video streaming. Additionally, we consider that the switches are heterogeneous in nature and have different TCAM capacity. We present the SD-DCN network as a flow network  $G(V, E)$ , where  $V$  and  $E$  denote the set of vertices and the set of edges in the flow network, respectively. Here, we consider that  $V = \bigcup_L V_I^L \cup \bigcup_L V_S^L$ , where  $V_I^L$  and  $V_S^L$  represent the set of IoT devices and the set of switches in the layer  $L$  of the flow network, respectively,  $L \geq 0$ , and  $V_S^0 = \{\emptyset\}$ . We consider that each edge  $e_{ij} \in E$  has a limited bandwidth  $B_{ij}$  and  $B_{ij} > 0$ , given that  $e_{ij} \neq 0$ . We consider two QoS parameters — throughput and delay — for evaluating the performance of the network. The throughput and delay per flow depend on the allocated capacity of the corresponding flow. In this work, we consider that the network flows are comprised of multiple *elephant* (high-volume) and *mice* (low-volume) flows [95]. Hence, we infer that there is a need to focus on ensuring both the optimal throughput and delay of the network. We consider that each IoT device  $n \in V_I^L$  generates  $F_n^E(t)$  and  $F_n^M(t)$  set of elephant and mice flows, respectively, at time instant  $t$ , where  $|F_n^E(t)|, |F_n^M(t)| \geq 0$ . Therefore, the number of flows  $F_L(t)$  in layer  $L$  of the network is as follows:

$$F_L(t) = \sum_{n \in V_I^L} [|F_n^E(t)| + |F_n^M(t)|] \quad (5.22)$$

We assume that there is sufficient space in the switches to place  $|F_L(t)|$  rules, i.e.,  $|F_L(t)| \leq \sum_{k \in V_S^{(L+1)}} R_k^{max}$ , where  $R_k^{max}$  is the maximum flow-rules which can be accommodated at SDN switch  $k$ . On the other hand, each elephant flow  $f_n^E(t) \in F_n^E(t)$  and each mice flow  $f_n^M(t) \in F_n^M(t)$  generate data with rate  $r_n^E(t)$  and  $r_n^M(t)$ , respectively. Therefore, the data-generation rate is given as  $\Psi(t) = \sum_{f_n^E(t) \in F_n^E(t)} r_n^E(t) + \sum_{f_n^M(t) \in F_n^M(t)} r_n^M(t)$ . Moreover, we consider that each SDN switch  $k \in V_S^{(L+1)}$  handles a set of installed rules  $R_k(t)$ , where  $R_k(t) \leq R_k^{max}$ . Therefore, we have:

## 5.2. FlowMan: The QoS-Aware Data Traffic Management Scheme

---

$$R_k(t) = |F_k^e(t)| + |F_k^m(t)| \quad (5.23)$$

where  $F_k^e(t)$  and  $F_k^m(t)$  represent the set of elephant and mice flow-rules installed in SDN switch  $k$ . Therefore, the throughput  $T_k(t)$  of SDN switch  $k$ , where  $\Psi(t) = \sum_{k \in V_S^{(L+1)}} T_k(t)$ , is defined as follows:

$$T_k(t) = \sum_{f_n^E(t) \in F_k^e(t)} r_n^E(t) + \sum_{f_n^M(t) \in F_k^m(t)} r_n^M(t) \quad (5.24)$$

Additionally, we consider that each SDN switch  $k$  has a processing capacity of  $\mu_k^S(t)$ . Considering that the flow processing at SD-DCN follows a packet-centric approach and following the work of Park and Schaar [91], the processing delay  $D_k(t)$  [96] is defined as follows:

$$D_k(t) = D_k^0 + \frac{\mu_k^S(t)}{T_k(t) - T_k^0(t)} \quad (5.25)$$

where  $T_k^0$  denotes the minimum resource share of each switch  $k$ ;  $D_k^0$  is the minimum delay for processing a packet, while considering the switch is idle, i.e., there is no packet in the switch. In this work, our objective is to obtain an optimal association among the data traffic flows and the switches for maximizing the network throughput and minimizing the delay.

### 5.2.2 Justification for Using Generalized Nash Bargaining Game

As mentioned earlier, the problem of the optimal distribution of heterogeneous data-flow among the available switches in SD-DCN to simultaneously ensure high throughput and low delay is an NP-hard problem, as it can be mapped to the zero-one knapsack problem [97]. To obtain a solution to this problem in polynomial time, we adopt a game-theoretic approach. In SD-DCN, the aforementioned problem scenario can be visualized

in the form of a “bargaining” situation in which the switches bargain among themselves to obtain their fair share of the data-flows. Here, the switches act cooperatively and try to agree upon a distribution of flow-rules which ensures mutual benefit. Moreover, in this work, the switches are also considered to be heterogeneous in terms of TCAM capacity. Hence, to model the considered problem scenario and the aforementioned properties of SD-DCN, we use the generalized Nash bargaining game. Here, the generalized Nash bargaining game not only takes into consideration the cooperative nature of the switches but also perfectly captures the heterogeneity of switches by introducing the Nash bargaining power of each switch. The Nash bargaining solution provides a Pareto optimal solution (which is shown in the subsequent sections) for deciding the maximum data-rate to be handled by each switch for ensuring high network throughput with less delay. Hence, we infer that the generalized Nash bargaining game is the most suitable technique for managing dynamic flow-traffic in the presence of multiple elephant and mice flows in SD-DCN.

### 5.2.3 Game Formulation

To study the interaction among the switches and the IoT devices in the considered problem scenario, we use *generalized Nash bargaining* game theory and propose a QoS-aware dynamic data traffic management scheme, named FlowMan, for SD-DCNs with IoT devices. Here, we consider that the switches act cooperatively to ensure high network throughput and low processing delay. In FlowMan, the strategy of each switch is to decide the optimal subset of flows to be handled by it. Based on the rule-space capacity of the switches, we introduce the bargaining power  $\alpha = \{\alpha_1, \dots, \alpha_k, \dots, \alpha_{|V_S^L|}\}$ , where  $\alpha_k$  denotes the bargaining power of switch  $k$ . Hence, we summarize the components of the generalized Nash bargaining game in FlowMan, as follows:

1. SD-DCN switches act as the players and bargain among themselves to distribute the set of flow rules to be installed.

## 5.2. FlowMan: The QoS-Aware Data Traffic Management Scheme

---

2. Data flows available in the network form the resource pool. Here, we only consider the flows generated/handled by one-hop neighbors, i.e., switches and IoT devices.
3. Switches are heterogeneous in terms of TCAM capacity.
4. The bargaining power of each switch depends on its TCAM capacity.

### 5.2.3.1 Utility Function

In the proposed FlowMan game, we consider that each switch  $k \in V_S^{(L+1)}$  decides its strategy, i.e., *bargain*, for distributing the one-hop network flows (including elephant and mice flows) while ensuring high network throughput and low delay in data traffic. The utility function  $\mathcal{U}_k(\cdot)$  of each switch  $k$  signifies the utilization of its capacity and TCAM memory. Motivated by the work of Park and Schaar [91], we consider that  $\mathcal{U}_k(\cdot)$  needs to ensure the following properties:

1. Each switch tries to maximize the utilization factor of its capacity. Therefore, each switch  $k$  tries to increase  $T_k(t)$ , while ensuring the constraint —  $R_k(t) \leq R_k^{max}$ .
2. Each switch  $k$  tries to reduce the overall network delay by reducing the packet queuing delay  $D_k(t)$ .

Therefore, we define  $\mathcal{U}_k(\cdot)$  for each switch as follows:

$$\mathcal{U}_k(T_k(t)) = \frac{\lambda}{D_k^0 + \frac{\mu_k^S(t)}{T_k(t) - T_k^0}} = \frac{\lambda(T_k(t) - T_k^0)}{D_k^0(T_k(t) - T_k^0) + \mu_k^S(t)} \quad (5.26)$$

where  $\lambda$  is a constant and  $\lambda > 0$ . Moreover, we consider that in FlowMan, each switch  $k$  ensures a minimum payoff  $d_k$  which is termed as its *disagreement point*. Disagreement point is calculated as the equilibrium point while the players act non-cooperatively. Here, we have —  $d_k = \mathcal{U}_k(T_k^0)$ . Hence, in FlowMan, we ensure that the payoff of each switch will be higher than  $\mathcal{U}_k(T_k^0)$ . Therefore, from Equation (5.26), we get —  $d_k = 0$ .

Considering that each switch  $k$  handles  $T_k(t)$  amount of resource, we define a *feasible utility set*  $\mathbf{S}$  as —  $\mathbf{S} = \{\dots, \mathcal{U}_k(T_k(t)), \dots\}$ . We get that  $\mathbf{S}$  is nonempty, convex, closed, and bounded. Additionally, we represent the disagreement point vector  $\mathbf{d}$  as —  $\mathbf{d} = \{\dots, \mathcal{U}_k(T_k^0), \dots\}$ . Here,  $\mathbf{d}$  is a set with elements having zero value. Thereby, the bargaining problem is defined as a tuple  $\langle \mathbf{S}, \mathbf{d} \rangle$ . We get that the Pareto optimal solution exists in FlowMan as defined in Definition 5.2.

**Definition 5.2.** We define that  $\mathbf{U}^*(t)$  is Pareto optimal, where  $\mathbf{U}^*(t) = (\dots, \mathcal{U}_k(T_k^*(t)), \dots)$ , iff we have —

$$(\dots, \mathcal{U}_k(T_k^*(t)), \dots) \geq (\dots, \mathcal{U}_k(T_k(t)), \dots) \quad (5.27)$$

As each switch has different bargaining power, we introduce the function  $F : \mathbf{S} \rightarrow \mathbb{R}^{|V_S^{(L+1)}|}$ , where we have —

$$\begin{aligned} F(\mathbf{S}, \mathbf{d}) &= \prod (\mathcal{U}_k(T_k(t)) - \mathcal{U}_k(T_k^0))^{\alpha_k} = \prod \mathcal{U}_k(T_k(t))^{\alpha_k} \\ &= \{\mathbf{u} \in \mathbf{B} \mid \mathbf{u} = \alpha \cdot \mathbf{T}(t), \sum \alpha_k = 1, \alpha_k \geq 0, \forall k\} \end{aligned} \quad (5.28)$$

where  $\mathbf{T}(t) = \{T_1(t), \dots, T_k(t), \dots, T_{|V_S^{(L+1)}|}(t)\}$  and  $\mathbf{B}$  is the bargaining set defined in Definition 5.3.

**Definition 5.3.** The bargaining set  $\mathbf{B}$  represents a set containing the Pareto optimal payoff pairs  $F(\mathbf{S}, \mathbf{d})$ , and  $\mathbf{B} \subseteq \mathbf{S}$ .

Hence, based on the feasible utility set  $\mathbf{S}$  and disagreement point vector  $\mathbf{d}$ , the optimization problem is defined as follows:

$$\max_{\mathbf{u} \in \mathbf{S}} F(\mathbf{S}, \mathbf{d}) \quad (5.29)$$

On the other hand, the solution  $\mathbf{U}^*$  needs to satisfy the following constraints —

$$\mathcal{U}_k(T_k(t)) > 0 \text{ and } \sum \alpha_k = 1, \text{ where } \alpha_k \geq 0, \forall k \quad (5.30)$$



### 5.2.4 Axioms for Generalized Nash Bargaining Solution

In this section, we examine the existence of the generalized Nash bargaining solution (please refer to Theorems 5.1-5.4) in the context of FlowMan, while ensuring that the following axioms are satisfied [91] — (1) Individual Rationality; (2) Feasibility; (3) Pareto Optimality; (4) Independence of Irrelevant Alternatives; and (5) Independence of Linear Transformations.

**Theorem 5.1.** *Feasible utility set  $\mathcal{S}$  is convex.*

*Proof.* We consider that the feasible utility set  $\mathcal{S}$  is convex, if for any  $\mathbf{u}^1, \mathbf{u}^2 \in \mathcal{S}$  and  $0 \leq \gamma \leq 1$ , we have —

$$\gamma \mathbf{u}^1 + (1 - \gamma) \mathbf{u}^2 \in \mathcal{S}$$

From Equation (5.24), we have —  $\sum T_k(t) = \Psi(t)$ . Therefore, we get:

$$\mathcal{S} = \left\{ \mathbf{u} \left| \sum \frac{\mu_k^S(t) \mathcal{U}_k(T_k(t))}{\lambda - D_k^0 \mathcal{U}_k(T_k(t))} = \Psi(t) \right. \right\} \quad (5.31)$$

Here, we observe that, for any  $T_k(t) > T_k^0$ , we have,  $\mathcal{U}_k(T_k(t)) > 0$ . Therefore, for the convexity, we need to have:

$$f(\gamma) \equiv \sum \frac{\mu_k^S(t) (\gamma \mathcal{U}_k^1(T_k(t)) + (1 - \gamma) \mathcal{U}_k^2(T_k(t)))}{\lambda - D_k^0 (\gamma \mathcal{U}_k^1(T_k(t)) + (1 - \gamma) \mathcal{U}_k^2(T_k(t)))} = \Psi(t) \quad (5.32)$$

Therefore, we have:

$$f(\gamma) = \begin{cases} \sum \frac{\mu_k^S(t) \mathcal{U}_k^2(T_k(t))}{\lambda - D_k^0 \mathcal{U}_k^2(T_k(t))}, & \text{if } \gamma = 0 \\ \sum \frac{\mu_k^S(t) \mathcal{U}_k^1(T_k(t))}{\lambda - D_k^0 \mathcal{U}_k^1(T_k(t))}, & \text{if } \gamma = 1 \end{cases} \quad (5.33)$$

On the other hand, to show that  $f(\gamma)$  is convex for  $0 < \gamma < 1$ , the second derivative of  $f(\gamma)$  needs to be non-negative. Therefore, we have:

$$\frac{d^2 f(\gamma)}{d\gamma^2} = \sum \frac{2\lambda \mu_k^S(t) D_k^0 (\mathcal{U}_k^1(T_k(t)) - \mathcal{U}_k^2(T_k(t)))^2}{[\lambda - D_k^0 (\gamma \mathcal{U}_k^1(T_k(t)) + (1 - \gamma) \mathcal{U}_k^2(T_k(t)))]^3} \quad (5.34)$$

where  $\mu_k^S(t) > 0$ ,  $D_k^0 > 0$ ,  $\lambda > 0$ , and  $(\cdot)^2 \geq 0$ . Additionally, we have,  $T_k(t) > T_k^0$ , therefore  $(\lambda - D_k^0 \mathcal{U}_k^1(T_k(t))) > 0$  and  $(\lambda - D_k^0 \mathcal{U}_k^2(T_k(t))) > 0$ . Thereby, we get that the denominator is greater than zero. In other words, the second derivative of  $f(\gamma)$  is non-negative. Hence,  $f(\lambda)$  is convex.  $\square$

**Theorem 5.2.** *In FlowMan, the function  $F(\mathbf{S}, \mathbf{d})$  satisfies Pareto optimality.*

*Proof.* We assume that there exists a point  $\mathbf{u} \in \mathbf{S}$  on the Pareto optimal front. However, from Equation (6)<sup>1</sup>, we get that  $\mathbf{u}^* > \mathbf{u}$ . Therefore, the assumption, that we made earlier, is not true. Hence, through contradiction, we prove that the function  $F(\mathbf{S}, \mathbf{d})$  satisfies Pareto optimality.  $\square$

**Theorem 5.3.** *The function  $F(\mathbf{S}, \mathbf{d})$  is independent of linear transformation.*

*Proof.* We consider that linear transformation is performed over the feasible utility set  $\mathbf{S}$ . Therefore, the utility function  $\mathcal{U}_k(T_k(t))$  of each switch  $k$  is transformed to  $\mathcal{U}_k(\beta_k T_k(t) + \rho_k)$ . Hence, we can rewrite Equation (5.29) as follows:

$$\begin{aligned} & \prod (\mathcal{U}_k(\beta_k T_k(t) + \rho_k) - \mathcal{U}_k(\beta_k T_k^0 + \rho_k))^{\alpha_k} \\ &= \prod \beta_k^{\alpha_k} \mathcal{U}_k(T_k(t))^{\alpha_k} \end{aligned} \tag{5.35}$$

Hence, we proof that the objective function is independent of the linear transformation.  $\square$

**Theorem 5.4.** *The function  $F(\mathbf{S}, \mathbf{d})$  is independent of irrelevant alternatives.*

*Proof.* We consider two feasible utility sets  $\mathbf{S}$  and  $\mathbf{S}'$ , where  $\mathbf{S}' \subset \mathbf{S}$ . Additionally, we consider that the Pareto optimal solution  $\mathbf{u}^*$  lies in both the set. Now, we assume that there exists another Pareto optimal solution  $\mathbf{u}'$  in the feasible utility set  $\mathbf{S}'$ . Therefore, we have —  $\mathbf{u}' > \mathbf{u}$ , which contradicts with the definition of Pareto optimal solution (refer to Definition 1<sup>2</sup>). Hence, we proof that the aforementioned assumption is not true, which concludes the proof.  $\square$

### 5.2.5 Existence of Generalized Nash Equilibrium

We prove the existence of generalized Nash equilibrium by using variational inequality (VI), as shown in Theorem 5.5.

**Theorem 5.5.** *Given a set of flows and the corresponding data-rates, there exists a generalized Nash equilibrium for each switch in the network.*

*Proof.* In FlowMan, we aim to maximize function  $F(\mathbf{S}, \mathbf{d})$ , which ensures cooperative benefit for the switches. Additionally, to prove that there exists a generalized Nash equilibrium, we need to show that the Hessian matrix of  $F(\mathbf{S}, \mathbf{d})$  is negative. We derive the Karush-Kuhn-Tucker (KKT) conditions, which are as follows:

1. Stationary condition:

$$\nabla F(\mathbf{S}, \mathbf{d}) - \nabla \sum \theta_k \mathcal{U}_k(T_k(t)) - \nu \nabla \left( \sum \frac{\mu_k^S(t) \mathcal{U}_k(T_k(t))}{\lambda - D_k^0 \mathcal{U}_k(T_k(t))} - \Psi(t) \right) = 0 \quad (5.36)$$

where  $\theta_k$  and  $\nu$  are Lagrangian multipliers.

2. Primal feasibility constraints:

$$\sum \frac{\mu_k^S(t) \mathcal{U}_k(T_k(t))}{\lambda - D_k^0 \mathcal{U}_k(T_k(t))} = \Psi(t) \quad (5.37)$$

$$\mathcal{U}_k(T_k(t)) > 0 \quad (5.38)$$

3. Dual feasibility condition:

$$\theta_k, \nu \geq 0 \quad (5.39)$$

4. Complementary slackness condition:

$$\sum \theta_k \mathcal{U}_k(T_k(t)) = 0 \quad (5.40)$$

Form *primal feasibility* and *complementary slackness* conditions, we get  $\theta_k = 0$ .

Hence, from *stationary* condition, we get that:

$$\nabla \mathcal{L} = \left[ \cdots, \alpha_k \frac{\prod_{l \neq k} \mathcal{U}_l(T_l(t))^{\alpha_l}}{\mathcal{U}_k(T_k(t))} - \frac{\nu \mu_k^S(t) \lambda}{(\lambda - D_k^0 \mathcal{U}_k(T_k(t)))^2}, \cdots \right]^T \quad (5.41)$$

Additionally, the Hessian matrix of  $F(\mathbf{S}, \mathbf{d})$  is negative matrix [91], which concludes the proof.  $\square$

### 5.2.6 Analysis of Generalized Nash Bargaining Solution

Based on the KKT conditions as mentioned in the Section 5.2.5, we have  $\nu \neq 0$ , where  $\nu$  is a Lagrangian multiplier. Therefore, we get that —

$$\nu = \frac{\alpha_k \mathcal{U}_k(T_k(t))^{(\alpha_k-1)} \prod_{l \neq k} \mathcal{U}_l(T_l(t))^{\alpha_l} (\lambda - D_k^0 \mathcal{U}_k(T_k(t)))^2}{\nu \mu_k^S(t) \lambda} \quad (5.42)$$

Here,  $\nu$  and  $\lambda$  are constants. Hence, we rewrite Equation (5.42) as follows:

$$\frac{\alpha_k (\lambda - D_k^0 \mathcal{U}_k(T_k^*(t)))^2}{\mu_k^S(t) \mathcal{U}_k(T_k^*(t))} = \frac{\alpha_l (\lambda - D_l^0 \mathcal{U}_l(T_l^*(t)))^2}{\mu_l^S(t) \mathcal{U}_l(T_l^*(t))} \quad (5.43)$$

where  $k \neq l$ , and  $k, l \in V_S^{(L+1)}$ . By replacing  $\mathcal{U}_k(T_k(t))$  with  $\frac{\lambda(T_k(t)-T_k^0)}{D_k^0(T_k(t)-T_k^0)+\mu_k^S(t)}$  (please refer to Equation (5.26)), we get:

$$\frac{\alpha_k \mu_k^S(t)}{(D_k^0 T_k(t) + \mu_k^S(t)) T_k(t)} = \frac{\alpha_l \mu_l^S(t)}{(D_l^0 T_l(t) + \mu_l^S(t)) T_l(t)} \quad (5.44)$$

We know that  $\sum T_k(t) = \Psi(t)$ . Hence, from Equation (5.44), we get that —

## 5.2. FlowMan: The QoS-Aware Data Traffic Management Scheme

$$T_l(t) = \frac{1}{2\alpha_k\mu_k^S(t)D_l^0} \left( -\alpha_k\mu_k^S(t)\mu_l^S(t) + \sqrt{\Phi_{kl}} \right) \quad (5.45)$$

where  $T_l(t) > 0$ ,  $D_l^0 > 0$ ,  $\zeta_k = (D_k^0 T_k(t) + \mu_k^S(t))$ , and

$$\Phi_{kl} = \left[ (\alpha_k\mu_k^S(t)\mu_l^S(t))^2 - 4\alpha_k\alpha_l\mu_k^S(t)\mu_l^S(t)\zeta_k T_k(t)D_l^0 \right].$$

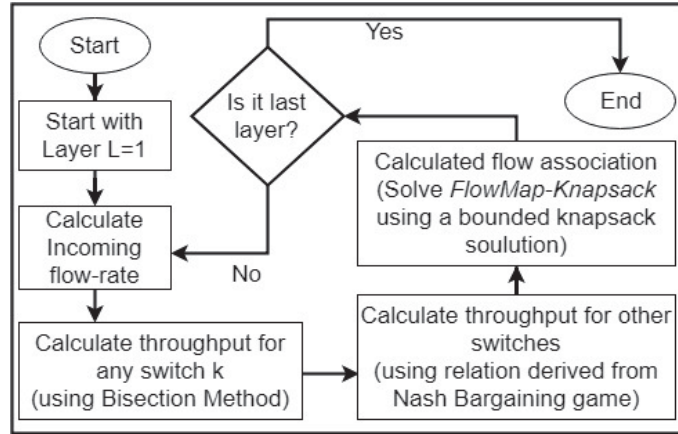


Figure 5.6: Workflow Diagram of FlowMan

### 5.2.7 Proposed Algorithm

From Section 5.2.6, we observe that, in SD-DCN, the controller can ensure efficient load distribution using FlowMan, while ensuring that the data flows are fairly distributed among the switches. The controller needs to calculate the optimal data rate  $T_k^*(t)$  for any switch  $k$  using a heuristic approach. In FlowMan, we use bisection method [98] having a fixed tolerance  $\epsilon > 0$ , as mentioned in Algorithm 5.2. Bisection method is used to evaluate the optimal value for  $T_k^*(t)$  as the upper and lower bounds of the utility functions are known for the time instant  $t$ . Thereafter, using Equation (5.45), which is derived based on the utility function  $\mathcal{U}_k(T_k(t))$ , the controller calculates the optimal data rate  $T_l^*(t)$  for each switch  $l$ , where  $k \neq l$ . Finally, the controller uses a bounded Knapsack solution method [99], named *FlowMan-Knapsack*, to decide the optimal distribution of data flows for each switch. The workflow diagram of FlowMan is presented in Figure

---

**Algorithm 5.2** Data Flow Management in FlowMan

---

**INPUTS:**

1:  $\Psi(t)$ ,  $\epsilon$ ,  $\cup_{n \in V^L} F_n^E(t)$ ,  $\cup_{n \in V^L} F_n^M(t)$ ,  $R_k^{max}$ ,  $\alpha$ ,  $V_S^{(L+1)}$ ,  $\mu^S(t)$ ,  $D^0(t)$

**OUTPUT:**

1:  $F_k^e(t)$ ,  $F_k^m(t)$ ,  $\forall k \in V_S^{(L+1)}$

**PROCEDURE:**

1:  $l \leftarrow 0$  and  $h \leftarrow \Psi(t)$   
2: **do**  
3:    $T_k(t) \leftarrow \frac{l+h}{2}$   
4:   **for** each  $l \in V_S^{(L+1)}$  and  $l \neq k$  **do**  
5:     Calculate  $T_l(t)$  using Equation (5.45) ▷ Obtained from the proposed Nash bargaining game-theoretic model  
6:   **end for**  
7:   **if**  $\sum T_k(t) \leq \Psi(t)$  **then**  
8:      $l \leftarrow T_k(t)$   
9:   **else**  
10:      $h \leftarrow T_k(t)$   
11:   **end if**  
12: **while**  $(h - l) < \epsilon$   
13:  $\mathcal{F}(t) \leftarrow (\cup_{n \in V^L} F_n^E(t)) \cup (\cup_{n \in V^L} F_n^M(t))$   
14: **for** each  $k \in V_S^{(L+1)}$  **do**  
15:    $F_k^e(t), F_k^m(t) \leftarrow \text{FlowMan-Knapsack}(T_k(t), \mathcal{F}(t))$   
16:    $\mathcal{F}(t) \leftarrow \mathcal{F}(t) / (F_k^e(t) \cup F_k^m(t))$   
17: **end for**  
18: **return**  $F_k^e(t)$ ,  $F_k^m(t)$ ,  $\forall k \in V_S^{(L+1)}$

---

## 5.2. FlowMan: The QoS-Aware Data Traffic Management Scheme

---

5.6.

### 5.2.7.1 Complexity Analysis

In FlowMan, two methods need to be executed sequentially. In the first method, i.e., the bisection method, the controller tries to find the optimal value for  $T_k(t)$ . Based on that, in the second method, the optimal data traffic associated with other switches are calculated using the bounded knapsack method. The complexity of the bisection method is  $O(|V_S^{(L+1)}|)$ , whereas, the complexity for bounded Knapsack method, i.e., FlowMan-Knapsack, is  $O(H|V_S^{(L+1)}|)$ , where  $H = \max\{R_k(t)|\forall k \in V_S^{(L+1)}\}$ . Hence, the overall complexity of FlowMan is  $O(H|V_S^{(L+1)}|)$ , as  $H \geq 1$ . Thus, FlowMan has a complexity of  $O(HC)$  for the overall network, where  $C = \max\{|V_S^L|, \forall L\}$ . Hence, we get that with the increases in the number of switches in each layer, the complexity of FlowMan increases. However, the complexity of FlowMan does not get affected by the increase in the number of layers in the network.

### 5.2.8 Performance Evaluation

In this section, we analyze the performance of FlowMan through simulation by varying the number of heterogeneous flows and the number of available switches.

#### 5.2.8.1 Simulation Parameters

We simulated FlowMan on Python3-based simulation platform. We considered that the switches and IoT devices are deployed randomly over the terrain of  $10 \times 10 \text{ km}^2$ , and each IoT device generates a single flow. We considered that each flow generates data traffic at a random rate. We considered that the threshold data rate is 0.1 million packets per second (*mpps*) to separate the flows into two categories – elephant and mice flows, as mentioned in Table 5.3. Additionally, we varied the number elephant flows, i.e., 5–10% of the total flows [95]. We also considered that each switch has an infinite buffer size.

We performed the simulation for 50 independent iterations. Each iteration is executed for 100 simulation seconds.

**Table 5.3:** Simulation Parameters

Parameter	Value
Number of SDN switches	5, 10, 15
Number of flows	5000, 10000, 20000
Number of elephant flows	5–10%
Data generation rate per mice flow	(0–0.1) <i>mpps</i>
Data generation rate per elephant flow	[0.1–0.5] <i>mpps</i>
Maximum flow rules /switch	4000–8000 [5]
Simulation duration	100 <i>sec</i>

### 5.2.8.2 Benchmarks

The performance of FlowMan is evaluated by comparing with existing schemes — FlowStat [100] and CURE [93]. In FlowStat, Bera *et al.* [100] proposed a flow-rule placement scheme based on the per-flow statistics. In this work, the authors tried to accommodate the maximum number of flows while finding an end-to-end path at once. We consider FlowStat as it deals with similar problems such as forwarding switch selection for heterogeneous flows and flow-rules placement to maximize the throughput of the network. On the other hand, in CURE, Maity *et al.* [93] proposed a scheme for flow management. The authors considered that the flow-rules are updated according to the priority of the switches. We consider CURE as it deals with the generic problems of rule placement in SD-DCN such as rule update while ensuring the high throughput of the network. In contrast to these existing schemes, FlowMan considers heterogeneous flows and aims to ensure high QoS, i.e., high network throughput and low delay, in SD-DCN. Additionally, FlowMan reduces an NP-hard problem to an NP-complete problem with the help of the generalized Nash bargaining game within finite time duration and ensures a Pareto optimal flow distribution among the switches in SD-DCN.



## 5.2. FlowMan: The QoS-Aware Data Traffic Management Scheme

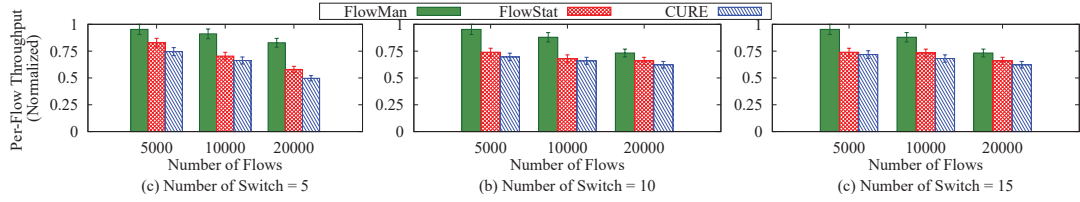


Figure 5.7: Per-Flow Throughput Analysis

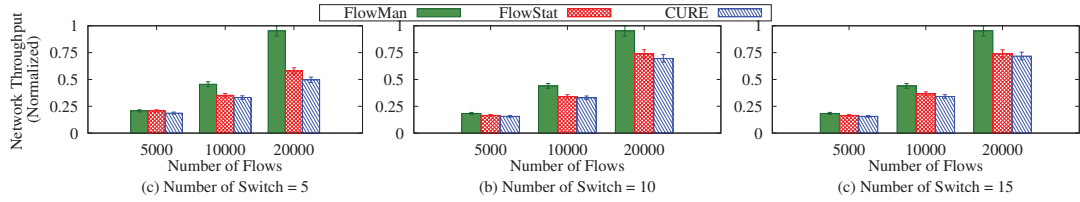


Figure 5.8: Network Throughput Analysis

### 5.2.8.3 Performance Metrics

We evaluated the performance of FlowMan using the following metrics:

**Per-Flow Throughput:** Per-flow throughput is calculated as the average amount of data processed for each flow over a certain duration, individually. In the presence of heterogeneous flows, per-flow throughput not only depends on the number of flow-rules installed at the switch but also on the data-rate associated with each flow.

**Network Throughput:** Network throughput is calculated as the amount of data processed cumulatively by the switches available in the network. It depends on the flow-association of each switch and the data-rate of each associated flow.

**Per-Flow Delay:** Per-flow delay is calculated as the queuing delay and the processing delay incurred by each flow at the associated switch. Due to the presence of heterogeneous flows, unbalanced data traffic results in high delay per-flow for the mice flows, when the elephant flows and mice flows are associated with the same switch.

**Network Delay:** Network delay is calculated as the end-to-end delay incurred by the flows available in the network. We claim that, with balanced load distribution, the network delay can be minimized significantly.

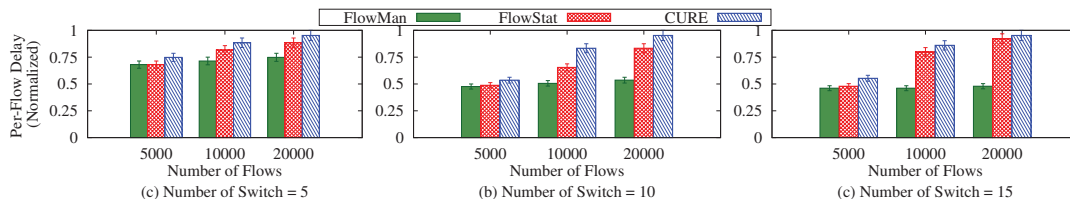


Figure 5.9: Per-Flow Delay Analysis

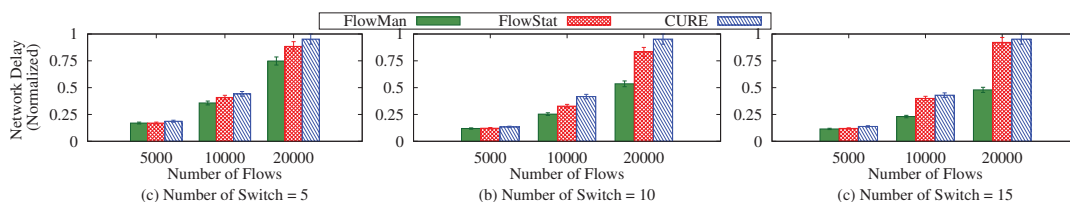


Figure 5.10: Network Delay Analysis

#### 5.2.8.4 Results and Discussions

From Figure 5.7, we observe that the per-flow throughput decreases with the increase in the number of flows, due to the increase in the number of elephant flows. However, using FlowMan, the per-flow throughput remains higher than using FlowStat and CURE, as FlowMan follows a Pareto optimal data-rate distribution for data flow management. We observe that FlowMan distributes the incoming flows with heterogeneous data-rate among the available switches, optimally. Hence, we yield that, using FlowMan, per-flow throughput increases by 24.6–47.8%. On the other hand, Figure 5.8 depicts that the network throughput increases significantly using FlowMan than using FlowStat and CURE. The network throughput depends on the elephant flows, as well as on the mice flows. We observe that, unlike FlowMan, the elephant flows get associated with a subset

### 5.3. Concluding Remarks

---

of switches using FlowStat and CURE, though these flows contribute almost 80% of the overall flow [95].

On the other hand, Figure 5.9 depicts that per-flow delay decreases by 27.7% using FlowMan than using FlowStat and CURE. In FlowMan, the traffic associated with each switch is optimal, hence there is no significant change in the delay of the associated flows. However, using FlowStat and CURE, the per-flow delay increases significantly because these schemes do not take into consideration the presence of heterogeneous flows. We yield that, FlowMan reduces the queuing delay by 77.8–98.7%. Similarly, Figure 5.10 depicts that the network delay increases exponentially with the increase in the number of flows. This is due to the fact that queuing delay has a significant impact on the overall network delay. Hence, we get that FlowMan ensures Pareto optimal flow distribution among the switches. Thereby, the queuing delay per-flow reduces significantly, which, in turn, reduces the overall network delay. Additionally, through theoretical analysis, we observe that using FlowMan, the flow-setup delay remains constant for a fixed set of switches. However, using the existing schemes – FlowStat and CURE, the flow-setup delay increases linearly with the increase in the number of flows in the network. Therefore, we observe that FlowMan enhances the performance of network universally, i.e., enhances the network QoS, while ensuring the per-flow QoS is maintained.

### 5.3 Concluding Remarks

In this Chapter, two dynamic data traffic management schemes in SD-DCN are presented considering that the IoT devices generate heterogeneous flows.

In this Chapter, we observed that dynamic data traffic management in SD-DCN is an NP-complete problem. Therefore, we formulated an evolutionary game theory-based TROD scheme to obtain a sub-optimal problem for data traffic management in SD-DCN in the presence of IoT-devices. We observed that TROD ensures proper distribution of data traffic among the switches while minimizing the network delay and maximizing the

## 5. QoS-Aware Data Traffic Management

---

throughput of the network. Moreover, through simulations, we observed that TROD outperforms the existing schemes – Mobi-Flow and CURE, while distributing of data traffic among the available switches and reducing the volumetric overhead per switch by 23.4-29.7%.

Thereafter, we yield that in the presence of heterogeneous flows, QoS-aware data flow management in SD-DCN is also NP-hard. Therefore, we proposed FlowMan to address the aforementioned problem. In FlowMan, we used the generalized Nash bargaining game to obtain a Pareto optimal data-rate distribution for the switches. Thereby, we achieved a sub-optimal problem which can be mapped to the bounded Knapsack problem, i.e., an NP-complete problem. Thereafter, we used a heuristic approach to solve the reduced sub-optimal problem. Additionally, we analyzed that FlowMan follows the axioms of the generalized Nash bargaining game. Through simulations, we observed that FlowMan outperforms the existing benchmark schemes — CURE and FlowStat, while ensuring high throughput and low delay. In particular, FlowMan reduces network delay by 77.8–98.7% and increases network throughput by 24.6–47.8%, than using the existing schemes.

## Chapter 6

# Broadcast Data Traffic Management

In this Chapter, we introduce a game-theoretic data broadcasting scheme, named D2B, for SD-DCN in the presence of mobile IoT devices. The bandwidth distribution among the devices at the edge-tier of the fat-tree SD-DCN follows a leader-follower structure. Hence, we use a single-leader-multiple-follower Stackelberg game for designing the D2B scheme. In fat-tree SD-DCN, each switch broadcasts big-data among the IoT devices and the servers at the edge-tier.

This Chapter is organized as follows. The system architecture considered in D2B scheme is discussed in Section 6.1. Section 6.2 focuses on the formulation of D2B data broadcasting scheme using single-leader-multiple-follower Stackelberg game. The algorithms proposed for D2B scheme are discussed in Section 6.3. We evaluate the performance of D2B in Section 6.4 while comparing with the existing schemes. Finally, Section 6.5 concludes this Chapter.

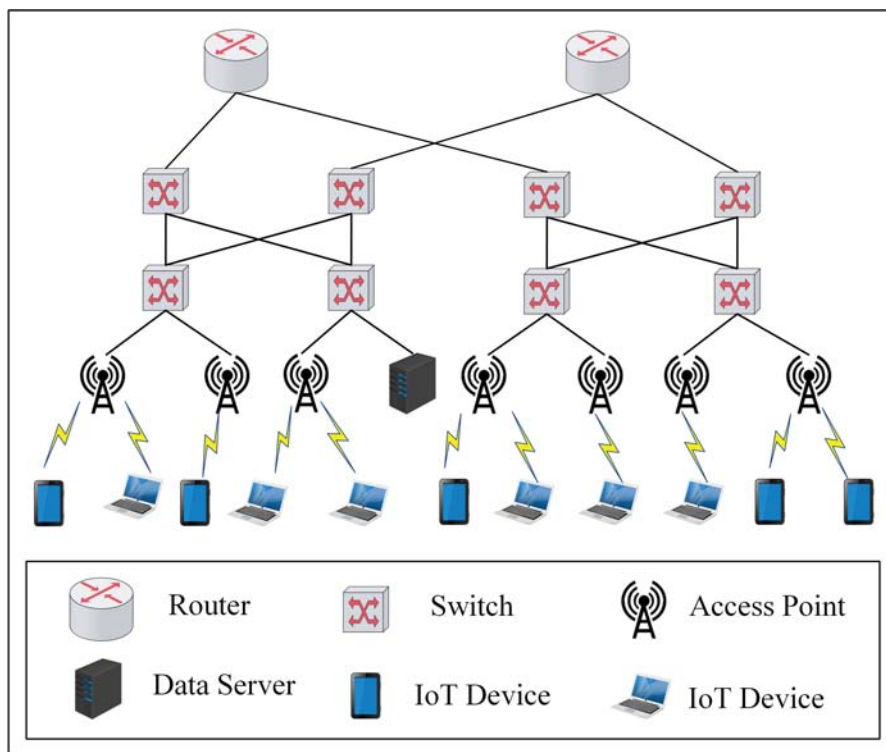


Figure 6.1: Schematic Diagram for Fat-Tree SD-DCN with IoT Devices

## 6.1 System Architecture

We consider an SD-DCN with fat-tree topology [21] in the presence of mobile IoT devices. A fat-tree topology is a three-tier network architecture having three tiers — *core*, *aggregation*, and *edge*. In SD-DCNs, fat-tree topology reduces blocking probability and is resilient to single-point failure due to the presence of multiple paths among any pair of nodes at the edge tier [21]. We consider that the mobile IoT devices are connected with switches at aggregation tier through Access Points (APs), as shown in Figure 6.1. In addition to the data-servers, each IoT device  $n \in \mathcal{A}_s \subseteq \mathcal{A}$ , where  $\mathcal{A}$  and  $\mathcal{A}_s$  denote the set of IoT devices available at the edge-tier and the set of IoT devices connected with switch  $s$ , respectively, at the edge-tier gets associated with a single switch  $s \in \mathcal{S}$  at the aggregation-tier, where  $\mathcal{S}$  represents the set of switches. We consider that the IoT

## 6.1. System Architecture

---

devices are owned by the end-users. On the other hand, the servers at the edge-tier are deployed by the network operators. The servers are used as storage devices only. We consider that these switches are static in nature and connected to specific routers at the core-tier. Additionally, we consider that the routers and the switches are deployed in a grid fashion over the terrain. Moreover, we consider that the complete coverage of IoT devices is ensured by the APs and the switches in the fat-tree SD-DCN.

Hence, to ensure throughput and delay-optimal big-data broadcast in the network from the source IoT device at the edge-tier, we need to allocate an optimal bandwidth to the IoT devices for downloading. Each IoT device  $n \in \mathcal{A}_s$  is connected with switch  $s$  at time instant  $t \in \mathcal{T}$ , where  $T$  is the set of time slots in a day. Each device  $n$  needs to decide the optimal data-rate  $r_n(t)$  (in *Kbps*), while satisfying the following constraints:

$$r_n^{min} \leq r_n(t) \leq r_n^{max} \quad \text{and} \quad r_n(t) \leq R_s^{max} - \sum \mathbf{r}_{-n} \quad (6.1)$$

where  $r_n^{max}$  and  $r_n^{min}$  denote the maximum and minimum data-rate requirement of device  $n$ ;  $R_s^{max}$  defines the capacity of switch  $s$  (in *Kbps*), and  $\mathbf{r}_{-n} \in \{r_1, \dots, r_{n-1}, r_{n+1}, \dots, r_{|\mathcal{A}_s|}\}$ .

On the other hand, each switch  $s$  tries to ensure the use of its bandwidth  $R_s^{max}$  for optimal throughput, and allocates bandwidth to the connected devices  $\mathcal{A}_s$ , while satisfying the constraints in Equation (6.1). Thus, the main challenges faced to develop the D2B scheme are as follows:

1. Modeling the D2B scheme, while considering the interaction between the IoT devices and the switches.
2. Developing algorithm for each device to decide the optimum downloading data-rate (in *Kbps*), while satisfying the constraints given in Equation (6.1).
3. Developing another algorithm for each switch  $s$  to decide the number of devices to serve at a time, while satisfying the constraint mentioned as  $\lim_{x \rightarrow 0^+} x < \epsilon$  and  $\lim_{x \rightarrow 0^-} x = 0$ , where  $0 < \epsilon \ll R_s^{max}$ , and  $x = [R_s^{max} - \sum_{n \in \mathcal{A}_s} r_n]$ . Hence, if switches

$s_1$  and  $s_2$  have the unused capacity  $x_{s_1}$  and  $x_{s_2}$ , respectively,  $x_{s_1} < x_{s_2}$  signifies that  $sf_{s_1} \prec sf_{s_2}$ , where  $sf_{s_1}$  and  $sf_{s_2}$  signify the satisfaction factor of switches  $s_1$  and  $s_2$ , respectively. We define the satisfaction factor of each switch  $s$  in Definition 6.1.

**Definition 6.1.** We define the satisfaction factor  $sf_s$  of switch  $s$  as ratio of the optimal availed throughput and the maximum capacity of switch  $s$ . Mathematically,

$$sf_s(t) = [\sum_{n \in \mathcal{A}_s} r_n(t)] / R_s^{max} \quad (6.2)$$

**Conjecture 6.1.** Based on Equation (6.1), we get that  $sf_s$  of each switch  $s$  follows constraint —  $sf_s \leq 1$ .

## 6.2 Proposed D2B Broadcast Scheme

To study the interaction between the switches and the IoT devices, we use a single-leader-multiple-follower Stackelberg game. This is a non-cooperative game, where each follower decides his/her/its strategy, non-cooperatively while satisfying the constraints imposed by the leader. In this Chapter, we divide the entire network into multiple blocks. In each block, an individual switch acts as the *leader*, and the devices, which are connected to the switch, act as *followers*. The proposed D2B scheme is formulated as a *pseudo-Cournot competition*, where each IoT device and the switch choose strategies, non-cooperatively and distributively. On the other hand, each switch distributes the available capacity among the connected IoT devices in order to achieve high performance with optimal throughput and delay for big-data broadcast in fat-tree SD-DCN. The components of D2B are as follows:

1. Each switch  $s$ , which acts as the leader, distributes the available bandwidth or capacity among the connected IoT devices, distributively.



## 6.2. Proposed D2B Broadcast Scheme

---

2. Each IoT device  $n$ , which acts as a follower, decides its downloading data-rate  $r_n$ , while satisfying Equation (6.1).
3. Each switch  $s$  tries to maximize its satisfaction factor, while utilizing the bandwidth capacity  $R_s^{max}$ .
4. There are  $\mathcal{M}$  chunks of data to be broadcasted by the source IoT device, where size of each chunk is  $m$  kb.
5. Each IoT device  $n$  and each switch  $s$  tries to maximize the payoffs of the utility functions  $\mathcal{U}_n(\cdot)$  and  $\mathcal{P}_s(\cdot)$ , respectively, in order to achieve throughput and delay-optimal broadcast in fat-tree SD-DCN.

**Definition 6.2.** *Pseudo cost coefficient  $p_s(t)$  of switch  $s$  at time instant  $t$  is defined as follows:*

$$p_s(t) = \sigma s f_s(t) \tag{6.3}$$

where  $\sigma$  is a constant.  $\sigma$  acts as a scaling factor and defines the variance of throughput of the switches in fat-tree SD-DCN.

### 6.2.1 Justification for Using Single-Leader-Multiple-Follower Stackelberg Game

The fat-tree SD-DCN follows a hierarchical architecture. In the fat-tree SD-DCN, the routers at the core tier and the switches at the edge tier are connected with wired links and the capacity of the links are fixed. On the other hand, the switches at the aggregation-tier take lead over the IoT devices and the servers at the edge-tier. Hence, we consider that the fat-tree SD-DCN follows a leader-follower architecture. Therefore, the IoT devices at the edge-tier behave non-cooperatively, and the fat-tree SD-DCN architecture follows the *pseudo-Cournot competition*. Additionally, each leader, i.e.,

each switch, decides its optimum strategy, distributively. Thereby, the throughput and delay-optimal big-data broadcasting in fat-tree SD-DCN in the presence of IoT devices is visualized as ‘*oligopolistic market*’. Hence, single-leader-multiple-follower Stackelberg game-theoretic approach is the most suitable approach for dynamic big-data broadcast in the presence of mobile IoT devices in fat-tree SD-DCNs, where the IoT devices at the edge-tier act non-cooperatively.

### 6.2.2 Utility Function of Each IoT Device

Using the utility function  $\mathcal{U}_n(\cdot)$ , each IoT device  $n \in \mathcal{A}$  finalizes the optimal data-rate  $r_n^*(t)$  at time instant  $t$ . The data-rate  $r_n(t)$  decided by each IoT device  $n$  depends on the data-rates  $r_{-n}(t)$  of the other IoT devices, indirectly. Thereby, each IoT device  $n$  decides data-rate  $r_n(t)$ , non-cooperatively. The utility function  $\mathcal{U}_n(\cdot)$  of each IoT device  $n$  needs to ensure the following properties:

1. Each IoT device  $n$  tries to download data with the maximum achievable data-rate. The utility function  $\mathcal{U}_n(\cdot)$  is considered to be non-decreasing function.
2. The utility function  $\mathcal{U}_n(\cdot)$  has a marginal value, which depends on  $r_n(t)$ . We represent the marginal condition of  $\mathcal{U}_n(\cdot)$  as follows:

$$\frac{\partial^2 \mathcal{U}_n(\cdot)}{\partial [r_n(t)]^2} < 0 \tag{6.4}$$

3. The pseudo cost coefficient  $p_s(t)$  has a negative influence on utility function  $\mathcal{U}_n(\cdot)$ . On the other hand, satisfaction factor  $sf_s(t)$  of each switch  $s$  varies proportionally with  $p_s(t)$ .

Therefore, we design the utility function  $\mathcal{U}_n(\cdot)$  as a concave function, which is represented as follows:

## 6.2. Proposed D2B Broadcast Scheme

---

$$\mathcal{U}_n(\cdot) = \beta \tan^{-1} \left( e^{-\frac{r_n(t) - r_n(t-\delta)}{r_n(t-\delta)}} \right) - p_s(t)r_n(t) \quad (6.5)$$

where  $\beta$  is a constant and  $\delta$  defines the time difference between two consecutive iterations. Each IoT device  $n$  tries to maximize its payoff value by deciding an optimal downloading data-rate, while satisfying constraints given in Equation (6.1). Hence, the objective of each device  $n$  is as follows:

$$\text{maximize } \mathcal{U}_n(\cdot) \quad (6.6)$$

### 6.2.3 Utility Function of Each Switch

For each switch  $s \in \mathcal{S}$ , we formulate the utility function  $\mathcal{P}_s(\cdot)$  for deciding the optimal throughput of the switch and minimize the network delay. Each switch  $s$  tries to maximize its satisfaction factor  $sf_s(t)$ , while utilizing the total capacity  $R_s^{max}$ . The pseudo price coefficient  $p_s(t)$  depends on  $sf_s(t)$ , as shown in Equation (6.3). Therefore, each switch  $s$  tries to maximize its payoff, while maximizing its utility function  $\mathcal{P}_s(\cdot)$ . Hence, the objective of each switch  $s$  is as follows:

$$\text{maximize } \mathcal{P}_s(\cdot) \quad (6.7)$$

We define the utility function  $\mathcal{P}_s(\cdot)$  of each switch  $s$  as multiplication of  $p_s(t)$  and  $sf_s(t)$ , where  $p_s(t)$  and  $sf_s(t)$  are defined in Equations (6.2) and (6.3), respectively. Mathematically,

$$\mathcal{P}_s(\cdot) = p_s(t)sf_s(t) \quad (6.8)$$

Hence, we observe that the utility function  $\mathcal{P}_s(\cdot)$  of each switch  $s$  follows a concave hyperbolic curve.

### 6.2.4 Existence of Equilibrium

We define the generalized Stackelberg-Nash equilibrium of D2B, as follows:

**Definition 6.3.** *The tuple  $\langle r_n^*(t), sf_s^*(t) \rangle$  is considered as the generalized Stackelberg-Nash equilibrium solution of switch  $s$ , if it satisfies the following inequalities:*

$$\begin{aligned} \mathcal{U}_n(r_n^*(t), \cdot, p_s^*(t)) &\geq \mathcal{U}_n(r_n(t), \cdot, p_s^*(t)) & (6.9) \\ \mathcal{P}_s(r_n^*(t), \mathbf{r}_{-n}^*(t), p_s^*(t), R_s^{max}) &\geq \mathcal{P}_s(r_n^*(t), \mathbf{r}_{-n}^*(t), p_s(t), R_s^{max}) \end{aligned}$$

where  $r_n^*(t)$  and  $sf_s^*(t)$  are the optimum data-rate decided by each IoT device  $n$  and the optimum satisfaction factor of switch  $s$ , respectively.

We ensure the existence of generalized Stackelberg-Nash equilibrium by using Variational Inequality (VI), as shown in Theorem 6.1. Moreover, in Section 6.2.5, we get the optimum concave solution under constraints given in Equation (6.1).

**Theorem 6.1.** *Given a fixed price coefficient  $p_s(t)$ , there exists a generalized Stackelberg-Nash equilibrium, as there exists a VI for the utility function  $\mathcal{U}_n(\cdot)$  of each IoT device  $n$ .*

*Proof.* In D2B, each IoT device  $n \in \mathcal{A}_s(t)$  tries to maximize its payoff at time instant  $t$ . Therefore, for the  $\mathcal{A}_s(t)$  set of IoT devices connected with the switch  $s$ , we define the overall utility function as follows:

$$\mathbf{u}_s(\cdot) = \sum_{n \in \mathcal{A}_s(t)} \mathcal{U}_n(t) \quad (6.10)$$

where  $\mathbf{u}_s(\cdot)$  must satisfy the constraints given in Equation (6.1). We evaluate Jacobian of matrix  $\mathcal{D}$ , where  $\mathcal{D} = \nabla \mathbf{u}_s(\cdot)$ , as follows:

## 6.2. Proposed D2B Broadcast Scheme

---

$$\mathcal{D} = \begin{bmatrix} \vdots & & \\ -\frac{\beta}{\left[2 + \left(\frac{r_n(t)}{r_n(t-\delta)}\right)^2\right] r_n(t-\delta)} - \frac{2r_n(t)}{R_s^{max}} - \frac{\sum \mathbf{r}_{-n}(t)}{R_s^{max}} & & \\ \vdots & & \end{bmatrix} \quad (6.11)$$

Thereafter, by neglecting  $\left[\frac{r_n(t)}{r_n(t-\delta)}\right]^2$  as  $\left[\frac{r_n(t)}{r_n(t-\delta)}\right]^2 \ll 1$ , we get Hessian matrix  $\nabla\mathcal{D}$  as follows:

$$\nabla\mathcal{D} = \begin{bmatrix} -\frac{2}{R_s^{max}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\frac{2}{R_s^{max}} \end{bmatrix} \quad (6.12)$$

Here, we observe that  $\nabla\mathcal{D}$  is a diagonal matrix, where each diagonal element is negative. Therefore, we conclude that D2B ensures the existence of Stackelberg-Nash equilibrium.  $\square$

### 6.2.5 Solution of Proposed D2B

For each IoT device  $n$ , by applying the KKT condition on utility function  $\mathcal{U}_n(\cdot)$  individually, we equate:

$$-\frac{\beta}{r_n(t-\delta) \left[ e^{\frac{\Delta r_n(t)}{r_n(t-\delta)}} \right]} - \frac{2r_n(t) + \sum \mathbf{r}_{-n}(t)}{R_s^{max}} = 0 \quad (6.13)$$

where  $\Delta r_n(t) = [r_n(t) - r_n(t - \delta)]$ . Hence, we get:

$$\begin{aligned} [\beta R_s^{max} - \sum \mathbf{r}_{-n}(t)] r_n(t - \delta) + 2r_n^2(t - \delta) \sum \mathbf{r}_{-n}(t) + r_n(t) [6r_n^2(t - \delta) + \sum \mathbf{r}_{-n}(t)] \\ - 4r_n(t - \delta) r_n^2(t) + 2r_n^3(t) = 0 \end{aligned} \quad (6.14)$$

Thereafter, using Cardano's method [101], we get:

$$r_n^*(t) = \sqrt[3]{-\frac{B}{2} + \sqrt{\frac{B^2}{2} + \frac{A^3}{27}}} - \sqrt[3]{\frac{B}{2} + \sqrt{\frac{B^2}{2} + \frac{A^3}{27}}} \quad (6.15)$$

where  $A = (\frac{c}{a} - \frac{b^2}{3a^2})$  and  $B = \frac{d}{a} + \frac{2b^3}{27a^3} - \frac{bc}{3a^2}$ ;  $a = 2$ ,  $b = -4r_n(t - \delta)$ ,  $c = 6r_n^2(t - \delta) + \sum \mathbf{r}_{-n}(t)$ , and  $d = [\beta R_s^{max} - \sum \mathbf{r}_{-n}(t)]r_n(t - \delta) + 2r_n^2(t - \delta) \sum \mathbf{r}_{-n}(t)$ .

For simplicity, we consider that the IoT devices are *homogeneous* in nature, i.e., the maximum data-rate that can be supported by the IoT devices is fixed. Hence, we get  $a = (|\mathcal{A}| + 1)$ ,  $b = -4r_n(t - \delta)$ ,  $c = 2(|\mathcal{A}| + 2)r_n^2(t - \delta) - (|\mathcal{A}| - 1)r_n(t - \delta)$ , and  $d = \beta R_s^{max} r_n(t - \delta)$ .

### 6.3 Proposed Algorithms for D2B

In order to reach the equilibrium in D2B, each IoT device and each switch decide their respective strategies for throughput and delay optimal big-data broadcast in fat-tree SD-DCN. Initially, each IoT device needs to be connected to a switch through an AP using Algorithm 6.1. Using Algorithm 6.1, each node selects the nearest switch and registers with that switch. This registration process needs to be repeated when that node comes to another region covered by a different switch. Thereafter, each IoT device decides and informs the optimum data-rate requirement to the concerned-switch using Algorithm 6.2 for downloading the broadcasted big-data. Using Algorithm 6.2, each IoT device initializes the downloading data rate to be minimum, and by maximizing its own utility function  $\mathcal{U}_n(\cdot)$ , IoT device  $n$  chooses an optimal downloading data-rate  $r_n^*(t)$ . On the other hand, using Algorithm 6.3, each switch decides an optimal pseudo price coefficient for maximizing the network throughput and minimizing the network delay. Based on the decided price coefficient, each IoT device tries to optimize the downloading data-rate, which indicates the throughput of the network. Moreover, the price coefficient depends proportionally on the number of IoT devices. Therefor, we get that if less number of IoT devices are associated with a switch, the delay at the switch reduces and the

### 6.3. Proposed Algorithms for D2B

---

throughput also decreases. On the other hand, if the number of IoT devices connected to a switch increases, the throughput increases, and the delay also increases. Hence, using Algorithms 6.2 and 6.3 sequentially, D2B tries to ensure a trade-off between the optimal network throughput and delay.

---

#### Algorithm 6.1 IoT Device Registration

---

**INPUT:**  $d_{ns}, \forall n \in \mathcal{A}, \forall s \in \mathcal{S}$

▷ Euclidean distance

**OUTPUT:**  $\{ \langle n, s \rangle, n \in \mathcal{A} \}$

**PROCEDURE:**

- 1: **for** each  $s \in \mathcal{S}$  **do**
  - 2:     Form a tuple of  $\langle n, s, d_{ns} \rangle$ ;
  - 3: **end for**
  - 4: Select the tuple having minimum  $d_{ns}$  value;
  - 5: **return**  $\{ \langle n, s \rangle, n \in \mathcal{A} \}$ ;
- 

---

#### Algorithm 6.2 Optimal Throughput for Each IoT Device $n$

---

**INPUTS:**

1:  $r_n(t - \delta), r_n(0) = 0, p_s^*(t), \beta$

2:  $\gamma$

▷ Data-rate increment factor in an iteration

**OUTPUT:**  $r_n^*(t)$

**PROCEDURE:**

- 1:  $r_n(t) = r_n^{min}$
  - 2: **while**  $\mathcal{U}_n(r_n^*(t), \cdot, p_s^*(t)) \geq \mathcal{U}_n(r_n(t), \cdot, p_s^*(t))$  **do**
  - 3:      $r_n(t) = r_n^*(t)$ ;
  - 4:     Evaluate the modified data-rate  $r_n^{mod}$  using Eq. (14)<sup>§</sup>;
  - 5:      $r_n^*(t) = r_n^{mod}$ ;
  - 6:     Call Algorithm 6.3;
  - 7: **end while**
  - 8: **return**  $r_n^*(t)$ ;
- 

### Complexity Analysis

In D2B, each IoT device registers with a switch using Algorithm 6.1. The computational complexity of Algorithm 6.1 is  $O(|\mathcal{S}|)$ . Thereafter, each IoT device  $n$  selects an optimal downloading data-rate using Algorithm 6.2. Considering that Algorithm 6.2 iterates  $K$  times before reaching Stackelberg equilibrium. Therefore, the computational complexity of Algorithm 6.2 is  $O(K)$ . For each iteration, Algorithm 6.3 having computational

---

**Algorithm 6.3** Optimal  $p_s(\cdot)$  for Each Switch  $s$

---

**INPUTS:**

- 1:  $\{r_n^*(t) | \forall n \in \mathcal{A}_s\}$ ,  $R_s^{max}(t)$

**OUTPUT:**  $p_s^*(t)$

**PROCEDURE:**

- 1:  $sf_s(t) = \sum_{n=1}^{\mathcal{A}_s} r_n^*(t)$ ;  
 2: Calculate  $p_s^*(t)$  using Eq. (5)<sup>§</sup>;  
 3: **return**  $p_s^*(t)$ ;
- 

complexity of  $O(1)$  is executed once. Therefore, the overall computational complexity of D2B is  $O(|\mathcal{S}| + K)$ .

## 6.4 Performance Evaluation

In this section, we analyze the performance of D2B through simulation by varying the number of heterogeneous IoT devices in terms of the bandwidth capacity.

### 6.4.1 Simulation Parameters

For the performance evaluation, we simulated using the MATLAB simulation platform. The IoT devices are deployed randomly over a terrain of  $1000 \times 1000 m^2$  [102]. However, the switches and the routers are deployed in a grid fashion, while ensuring full coverage. We considered that the source IoT device generates 1000 number of data chunks, and the size of each data chunk is 800 *Mb*, as shown in Table 6.1. Motivated by the device distribution of the Internet [13, 103], we considered that the distribution of IoT device capacities follows the distribution mentioned in Table 6.2 [103].

### 6.4.2 Benchmarks

The performance of D2B is evaluated while comparing with two existing schemes for SD-DCNs — the Lock-Step Broadcast Tree based big-data broadcasting (LSBT) [13] and the Multicast Fat-Tree Data Center Networks (DCN\_INFOCOM) [21] schemes. In



## 6.4. Performance Evaluation

---

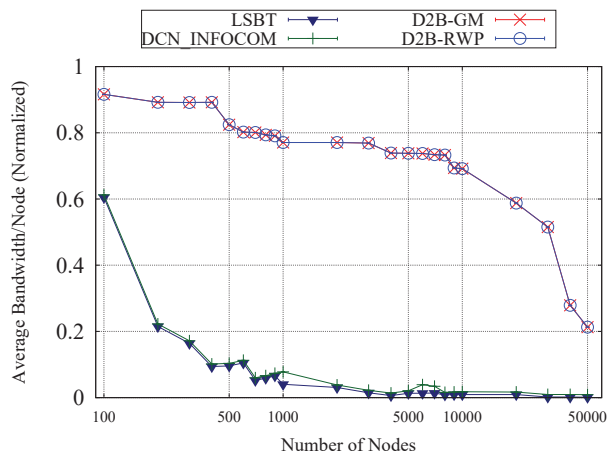
**Table 6.1:** Simulation Parameters

Parameter	Value
Simulation Area	1000 $m \times$ 1000 $m$ [102]
Number of Nodes	100 – 50000
Number of Switches	4
Number of Servers	3
Capacity of Nodes	128, 384, 1000, 5000 $Kbps$
Velocity of Source Node	5 $m/s$
Capacity of Switches	10 $Gbps$
Data chunks generated	1000
Size of each data chunk	800 $Mb$
Mobility model (MM)	Random Gauss-Markov [104] Random waypoint [105]

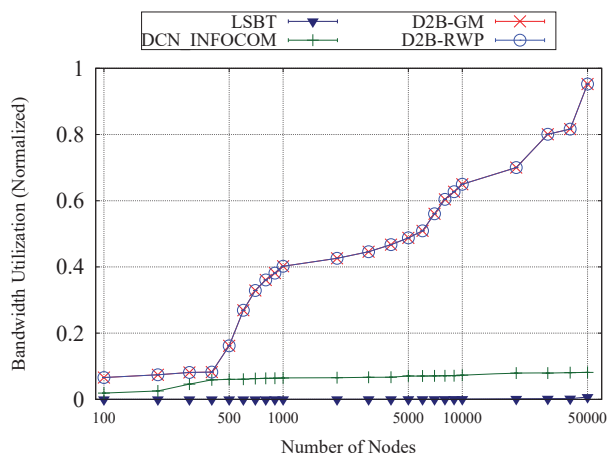
**Table 6.2:** Node Capacity Distribution

Capacity ( $Kbps$ )	Nodes (%)
128	20
384	40
1000	25
5000	15

LSBT, Wu *et al.* [13] proposed a big-data broadcasting scheme, while forming a *Lock Step Broadcast Tree* which is considered as basic unit of upload bandwidth. The authors also considered that the source device, which has the maximum capacity in the network, is at the root of the tree. On the other hand, in DCN\_INFOCOM, Guo and Yang [21] proposed a fat-tree based SD-DCN. In DCN\_INFOCOM, the authors tried to minimize the number of core switches needed to overcome the problem of over subscriptions. Additionally, the authors overlooked the problem of balanced bandwidth distribution. Moreover, these works do not consider the presence of the mobile IoT devices in fat-tree SD-DCN. In the presence of IoT devices in fat-tree SD-DCN, we improve the network performance for big-data broadcast, while ensuring optimal throughput and delay of the network using D2B. Moreover, we simulated D2B with two mobility models — random Gauss-Markov [104] and random waypoint mobility [105], and named the schemes as D2B-GM and D2B-RWP, respectively.



**Figure 6.2:** Average Bandwidth Allocation per Node



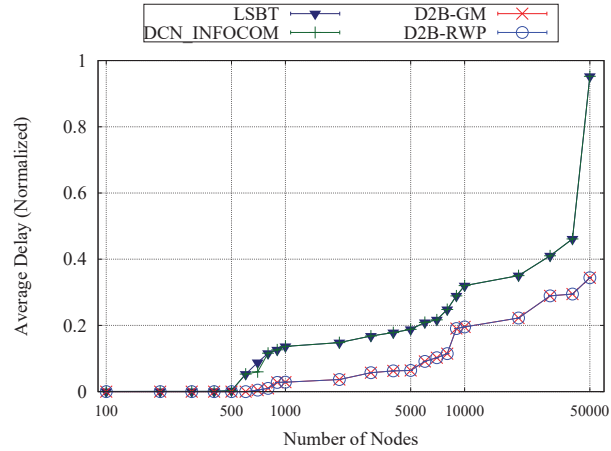
**Figure 6.3:** Total Bandwidth Utilization

### 6.4.3 Performance Metrics

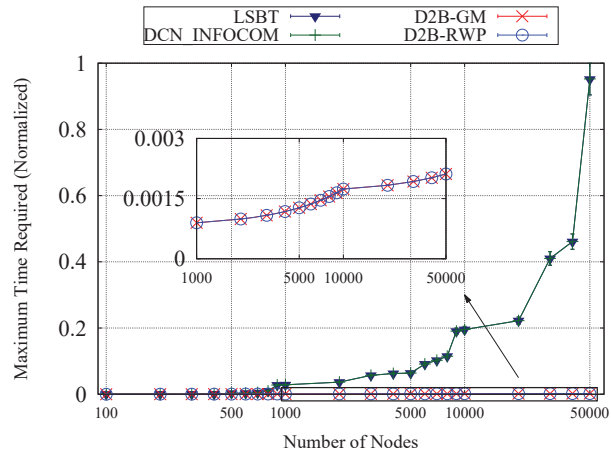
We have evaluated the performance of D2B using the following metrics:

**Bandwidth Utilization:** We consider that the IoT devices are heterogeneous in nature. Additionally, these devices are connected with the switches having limited bandwidth. Hence, we calculate the bandwidth utilization factor of each IoT device as a ratio of bandwidth usage for big-data broadcast and the maximum capacity of the IoT device.

## 6.4. Performance Evaluation



**Figure 6.4:** Average Delay of the Network



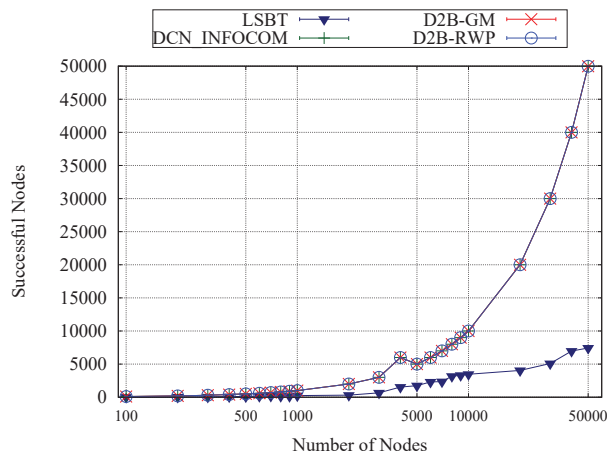
**Figure 6.5:** Maximum Time Required for Broadcasting 100 Packets

**Network Delay:** We define network delay as the total time required to complete the big-data broadcast in the fat-tree SD-DCN. Hence, the network delay is defined as the time duration needed for completion of data reception by all the IoT devices in the fat-tree SD-DCN.

**Successful Nodes:** We consider an IoT device as a successful node if that IoT device receives all the broadcasted data packets sent by the source IoT device, successfully.

### 6.4.4 Results and Discussions

Figures 6.2 and 6.3 show that the bandwidth utilization increases using D2B than using LSBT and DCN\_INFOCOM. We observe that D2B yields 33-55% increase in the average amount of bandwidth allocated per IoT device. In LSBT, big-data is broadcasted from the main server having higher network capacity. In DCN\_INFOCOM, the allocation of bandwidth is done sequentially. On the other hand, using D2B, the bandwidth is allocated per IoT device, distributively. Hence, using D2B, bandwidth utilization per IoT device is higher than using other schemes — LSBT and DCN\_INFOCOM. Additionally, from Figure 6.3, we observe that the overall network bandwidth utilization increases by at least 55.32% using D2B than using LSBT and DCN\_INFOCOM.



**Figure 6.6:** Successful Nodes in Broadcasting

From Figure 6.4, we observe that, using D2B, average delay decreases 25.83-62.4% than using LSBT and DCN\_INFOCOM. In D2B, due to an increase in the average bandwidth allocated per IoT device, the overall network delay decreases. Moreover, from Figure 6.5, we observe that using D2B, with the increase in the number of devices, the total delay in data broadcasting increases linearly, whereas using LSBT and DCN\_INFOCOM, the total time required to complete the process increases exponentially. From Figure 6.6, we observe that the number of IoT devices, which received

## 6.5. Concluding Remarks

---

broadcasted data packets successfully, is comparable using D2B and DCN\_INFOCOM. However, using LSBT, the source IoT device is considered to be at the root of the tree. Hence, using LSBT, the IoT devices having a lesser capacity than the source IoT device form the subtree, which includes the successful nodes. Thereby, using LSBT, the number of IoT devices, which are successful in receiving the broadcasted packets, is lesser than using D2B and DCN\_INFOCOM. Moreover, we observe that the bandwidth distribution using D2B is temporal. Hence, we observe that in Figures 6.2-6.6, the results for D2B-GM and D2B-RWP are almost similar. Thereby, we conclude that D2B ensures efficient distribution of available bandwidth among the connected IoT devices. Hence, we conclude that D2B ensures dynamic big-data broadcast in fat-tree SD-DCN in the presence of mobile IoT devices with optimal throughput and network delay.

## 6.5 Concluding Remarks

In this Chapter, we formulated a single-leader-multiple-follower Stackelberg game theory-based D2B scheme to ensure proper bandwidth utilization of the network for dynamic big-data broadcast in fat-tree SD-DCN in the presence of heterogeneous mobile IoT devices. We observed that D2B ensures the reduction in network delay in the presence of the mobile IoT devices at the edge-tier of the fat-tree SD-DCN. Moreover, from simulation, we observe that D2B outperforms the other existing schemes — LSBT and DCN\_INFOCOM. In particular, we observed that using D2B, the network throughput increased by 55.32%, while ensuring at least 33% increase in the average bandwidth allocation per IoT device, and reduction in the overall delay.



## Chapter 7

# Multicast Data Traffic Management

In this Chapter, we use a single-leader-multiple-follower Stackelberg game for designing the dynamic data multicasting scheme, named D2M, in SD-DCN. In D2M, the controller takes strategic decision for rule placement among the switches, while ensuring efficient load balancing. On the other hand, the switches are responsible for processing the data packets. Additionally, we consider that the switches are capable of bisecting the capacity (bandwidth), and a fraction of the available bandwidth is to be allocated for each flow. The allocated bandwidth for each flow is not shared with other flows and remains dedicated, if there is no change in network. Hence, in D2M, we consider that the controller acts as the leader and the switches act as the followers.

This Chapter is organized as follows. The system architecture considered in D2M scheme is discussed in Section 7.1, while mentioning the assumptions. Section 7.2 focuses on the formulation of D2M using single-leader-multiple-follower Stackelberg game. The algorithms proposed for D2M scheme are discussed in Section 7.3. We evaluate the performance of D2M in Section 7.4 while comparing with the existing schemes. Finally, Section 7.5 concludes this Chapter.

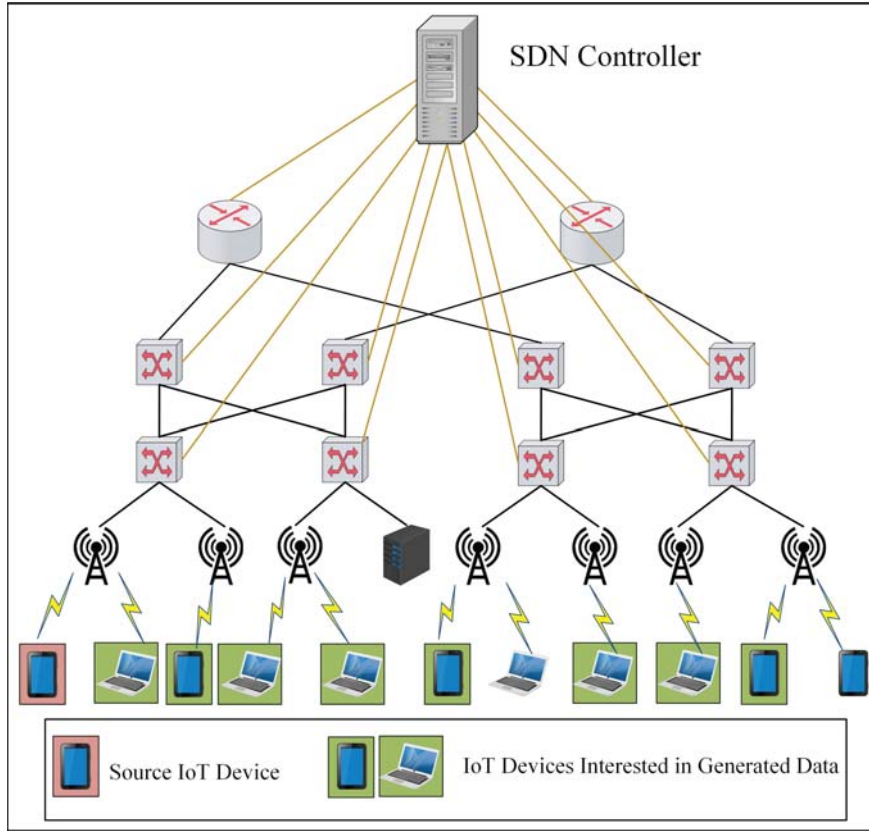


Figure 7.1: Schematic Diagram of Fat Tree-based SD-DCN with IoT Devices

## 7.1 System Model

In this Chapter, we consider an SD-DCN, where DCN follows the fat-tree architecture. A fat-tree is a network architecture having three tiers — *core*, *aggregation*, and *edge*. The routers at the core-tier are responsible for connecting the intra-networks. We consider that the routers have enough bandwidth to support the connected switches at the aggregation-tier, as shown in Figure 7.1. On the other hand, the switches are responsible for providing communication services to the end-devices at the edge-tier. Additionally, we consider that the switches and the routers in the fat-tree DCN have limited ternary content-addressable memory (TCAM). In other words, the flow-tables have limited number of flow-entries. We consider that each switch  $s \in \mathcal{S}$ , where  $\mathcal{S}$  is the set of switches,



## 7.1. System Model

---

provides services to  $\mathcal{F}_s(t)$  set of heterogeneous flows at time instant  $t$ . Considering that each IoT device generates at most one flow at time instant  $t$ , we have —  $|\mathcal{A}_s(t)| = |\mathcal{F}_s(t)|$ , where  $\mathcal{A}_s(t) \subseteq \mathcal{A}$ , and  $\mathcal{A}_s(t)$  and  $\mathcal{A}$  denote the set of active devices in data generation and the available heterogeneous IoT devices, respectively. These heterogeneous IoT devices are mobile in nature and connected wirelessly with the switches at the aggregation-tier through access points (APs). In addition to the IoT devices, the edge-tier includes data-servers. These data-servers are connected with the switches at the aggregation-tier using a wired connection, as shown in Figure 7.1.

Additionally, we consider that each SDN switch  $s \in \mathcal{S}$  has a capacity of  $C_s$  (in *kbps*). We consider that each flow  $f_n \in \cup \mathcal{F}_s(t)$ , where  $n \in \mathcal{A}$ , has a minimum data-rate requirement which is denoted as  $r_n^{min}$  (in *kbps*). The maximum data-rate of the flows generated by the heterogeneous IoT devices are constrained by hardware limitations. We denote the maximum data-rate of flow  $f_n$  as  $r_n^{max}$  (in *kbps*). Hence, the actual data-rate  $r_n(t)$  (in *kbps*) of each flow  $f_n$  has to satisfy the following constraint:

$$r_n^{min} \leq r_n(t) \leq r_n^{max} \quad (7.1)$$

Therefore, at time instant  $t$ , the number of IoT devices which are getting served by each switch  $s$ , needs to satisfy the following constraint:

$$C_s^{rem} \geq \sum_{f_n \in \mathcal{F}_s(t)} r_n(t), \quad \forall s \in \mathcal{S} \quad (7.2)$$

where  $C_s^{rem}$  is the remaining capacity of switch  $s$  after allocating bandwidth to the data center servers. Considering that each flow  $f_d$  associated with data-server  $d \in \mathcal{D}$ , where  $\mathcal{D}$  is the set of data-servers in the SD-DCN, handles data-rate  $r_d(t)$ , we define  $C_s^{rem}$  as follows:

$$C_s^{rem} = C_s - \sum_{f_d \in \mathcal{F}_s(t)} r_d(t) \quad (7.3)$$

We consider that a set IoT devices  $\bar{\mathcal{A}}(t) \subseteq \mathcal{A}$  generates a set of flows  $\bar{\mathcal{F}}(t)$  at time instant  $t$ . Therefore, we have  $\bar{\mathcal{F}}(t) = \bigcup_s \mathcal{F}_s(t)$ . Additionally, we define an *associative* variable  $x_{ns}$ , which is defined as follows:

$$x_{ns} = \begin{cases} 1, & \text{if } f_n \in \mathcal{F}_s(t) \\ 0, & \text{otherwise} \end{cases} \quad (7.4)$$

**Proposition 7.1.** *For each flow  $f_n \in \bar{\mathcal{F}}(t)$ , the following condition is true:*

$$1 \leq \sum_{s \in \mathcal{S}} x_{ns} \leq 4 \quad (7.5)$$

*Proof.* In SD-DCN, we consider that flow-rule associated with each flow  $f_n$  needs to be installed at least *one* SDN switch in case of intra-pod communication. On the other hand, the flow  $f_n$  needs to be matched at most *four* intermediate SDN switches, as we have considered two-tier fat-tree architecture. Therefore, we proof that the condition mentioned in Equation (7.5) is true.  $\square$

### 7.1.1 Assumptions

While designing D2M, we assume that:

1. We consider that each IoT device is always connected with one of the access points available in SD-DCN.
2. We consider that the centralized controller controls the flow-rules to be installed at the switches.
3. The controller can change its strategy of choosing an optimal source node at any given instant if it finds a source node with high payoff.
4. Each IoT device, which is downloading data, is interested in a single flow.

## 7.2 Dynamic Data Multicasting (D2M) Scheme

### 7.2.1 Game Formulation

For modeling the interaction between the SDN switches and the controller, we use single leader multiple follower Stackelberg game. In D2M, we consider that the controller acts as the leader and installs the flow-rules in the SDN switches. Additionally, the controller decides the source node of the flow for each destination. On the other hand, the SDN switches, which act as the followers, decide their respective strategy, non-cooperatively. The followers help the controller to manage the network properly while deciding the amount of bandwidth to be allocated for each flow and optimize the usage of overall capacity. D2M is formulated as a *pseudo-Cournot competition*. The components in the D2M are as follows:

1. The controller acts as the leader. It decides the optimal route of for each flow and flow-rules to be installed to which switches.
2. Each SDN switch act as a follower. The switches make a trade-off between the associated flow-rules and the bandwidth allocated for each flow.
3. Each switch tries to maximize its satisfaction factor, which is defined in Definition 7.1 while maximizing the network throughput.
4. We consider that each flow  $f_n$ , which is generated by IoT device  $n$ , comprises of  $M_n$  amount of data (in Kb).

**Definition 7.1.** *The satisfaction factor  $\rho_s(t)$  of each switch  $s$  is defined as the ratio of amount of bandwidth allocated to  $\mathcal{F}_s(t)$  flows and the capacity  $C_s$  of the switch. Mathematically,*

$$\rho_s(t) = \frac{\sum_{f_n \in \mathcal{F}_s(t)} r_n(t) + \sum_{f_d \in \mathcal{F}_s(t)} r_d(t)}{C_s} \quad (7.6)$$

### 7.2.1.1 Justification for Single Leader Multiple Follower Stackelberg Game

In SDN, the tasks – network control and packet processing – are divided into two planes – control and data planes, respectively. The switches contain the data plane and inform the controller if there is a table-miss. On the other hand, the controller, which is associated with the control plane, takes care of the flow routing and updates the flow-tables at the switches. Hence, we consider that SD-DCN follows a leader-follower architecture. Thereby, in order to model the interactions among the controller and the switches, we consider single leader multiple follower Stackelberg game theoretic approach. In D2M, the controller decides the optimal routing path among the source-destination pair, for ensuring delay-optimal data multicasting. Thereafter, the switches decide the amount of bandwidth to be allocated for each flow while ensuring the throughput-optimal data multicasting. Hence, data multicasting in SD-DCN resembles an oligopolistic market scenario. Hence, we infer that a single leader multiple follower Stackelberg game is the most suitable approach for data multicasting in SD-DCN.

### 7.2.1.2 Utility Function of Controller

The utility function  $\mathcal{U}_c(\cdot)$  of the controller signifies the overall network delay for multicasting while maximizing the network lifetime. The controller decides the source node for each flow in multicasting and the routing path while maximizing the payoff value of utility function  $\mathcal{U}_c(\cdot)$ . The utility function  $\mathcal{U}_c(\cdot)$  needs to satisfy the following properties:

1. With the increase in hop-count, the payoff value decreases.
2. With the increase in residual energy of the source IoT device, payoff value increases.
3. If the source node selected for a flow associated with a destination is serving other flow(s) associated with other destination(s), the payoff value increases.

Therefore, we consider that the utility function  $\mathcal{U}_c(\cdot)$  of the controller is a linear function and is defined as follows:

## 7.2. Dynamic Data Multicasting (D2M) Scheme

---

$$\mathcal{U}_c(\cdot) = \sum_{s \in \mathcal{S}} \sum_{f_n \in \mathcal{F}_s(t)} \left( \alpha_n \frac{E_n^{res}(t)}{E_{max}} + \beta_n \frac{h_{f_n}(t)}{4} + \gamma_n a_n(t) \right) \quad (7.7)$$

where  $E_{max}$  and  $E_n^{res}(t)$  denote the maximum and residual energy of IoT device  $n$  at time instant  $t$ ;  $h_{f_n}(t)$  denotes the hop-count associated with flow  $f_n$ ; and  $a_n(t)$  is a binary variable and denotes the state of IoT device  $s$ . We defined  $a_n(t)$  in Definition 7.2. In Equation (7.7),  $\alpha_n$ ,  $\beta_n$ , and  $\gamma_n$  are constants specified for flow  $f_n$ . These constants ensure that the associated coefficients have similar numeric impact on the payoff value of utility function  $\mathcal{U}_c(\cdot)$ . The controller tries to maximize  $\mathcal{U}_c(\cdot)$ , while ensuring the constants mentioned in Equations (7.2) and (7.5).

**Definition 7.2.** *We consider that each IoT device  $n$  have two states — idle and active. Additionally, we consider that the IoT devices at idle state has zero energy consumption, whereas the energy consumption of the IoT devices in active state is finite and positive. Therefore, the binary variable  $a_n(t)$  of IoT device  $n$  is defined as follows:*

$$a_n(t) = \begin{cases} 1, & \text{if IoT device } n \text{ is in active state} \\ 0, & \text{otherwise} \end{cases} \quad (7.8)$$

### 7.2.1.3 Utility Function of Each Switch

The utility function  $\mathcal{U}_s(\cdot)$  of each switch  $s$  signifies the utilization of capacity of the switch, and the flow-associated delay. Each switch decides the amount of bandwidth to be allocated to each flow  $f_n \in \mathcal{F}_s(t)$ , while maximizing the payoff value of the utility function  $\mathcal{U}_s(\cdot)$ . Hence,  $\mathcal{U}_s(\cdot)$  must satisfy the following properties:

1. With the increase in the satisfaction factor  $\rho_s(t)$ , the payoff value increases.
2. With the increase in each flow-associated delay, the payoff value decreases.
3. With the increase in overall delay associated with  $\mathcal{F}_s(t)$  flows, the payoff value decreases.

Therefore, we define the utility function  $\mathcal{U}_s(\cdot)$  as follows:

$$\mathcal{U}_s(\cdot) = \zeta_s \sum_{f_n \in \mathcal{F}_s(t)} \frac{r_n(t)}{r_n^{max}} + \phi_s \sum_{f_n \in \mathcal{F}_s(t)} \frac{M_n}{r_n(t)} + \varphi_s \rho_s(t) \quad (7.9)$$

where  $\zeta_s$ ,  $\phi_s$ , and  $\varphi_s$  are constants for switch  $s$ . These constants ensure that the associated coefficients have similar numeric impact on the payoff value of utility function  $\mathcal{U}_s(\cdot)$ . The first and second terms in Equation (7.9) signify the utilization of capacity per flow and the flow-associated delay, respectively. Hence, each switch aims to decide an optimal data-rate for each flow  $f_n \in \mathcal{F}_S(t)$ , while maximizing the payoff value of  $\mathcal{U}_s(\cdot)$  and satisfying the constraints mentioned in Equations (7.1) and (7.2).

### 7.2.2 Existence of Nash Equilibrium

We consider the Nash equilibrium of D2M, as defined in Definition 7.3. We get that at the Nash equilibrium point, the players cannot increase their payoff value by deviating from the equilibrium point.

**Definition 7.3.** *We define the Nash equilibrium point as a tuple of  $r_n^*(t)$  and  $n^*$ , where  $n^*$  and  $r_n^*(t)$  represent the optimum source node for flow  $f_n$  and the optimum bandwidth allocated to IoT device  $n^*$ , respectively. The Nash equilibrium point needs to satisfy the following constraints:*

$$\begin{aligned} \mathcal{U}_c(E_{n^*}^{res}(t), h_{f_{n^*}}(t), a_{n^*}(t), E_{-n^*}^{res}(t), h_{f_{-n^*}}(t), a_{-n^*}(t)) \geq \\ \mathcal{U}_c(E_n^{res}(t), h_{f_n}(t), a_n(t), E_{-n^*}^{res}(t), h_{f_{-n^*}}(t), a_{-n^*}(t)) \end{aligned} \quad (7.10)$$

$$\mathcal{U}_s(r_n^*(t), r_{-n^*}^*(t)) \geq \mathcal{U}_c(r_n(t), r_{-n}^*(t)) \quad (7.11)$$

where  $a_{-n^*}(t) = \{a_{k^*}(t) | \forall f_k \in \overline{\mathcal{F}}(t) \text{ and } k \neq n\}$ . Similarly, we can define  $E_{-n^*}^{res}$  and

## 7.2. Dynamic Data Multicasting (D2M) Scheme

---

$h_{f_{-n}^*}(t)$ . On the other hand,  $r_{-n}^*(t) = \{r_p^*(t) | \forall p \in \mathcal{F}_s(t) \text{ and } p \neq n\}$

We get that D2M follows a finite perfect information game theoretic approach. Additionally, the players in D2M follow pure strategies. Therefore, using the backward-induction method, we can ensure that there exists at least one Nash equilibrium point in D2M [106].

### 7.2.3 Theoretical Analysis of D2M Scheme

The controller selects its optimal strategy based on preference relation of the available strategies. For example, we consider that there are two flows  $f_1$  and  $f_2$ , and IoT devices  $n'$  and  $n''$  have the data for multicasting, i.e., these devices are probable source nodes. We consider that the controller has a preference relation —

$$(f_1 \rightarrow n', f_2 \rightarrow n') \succeq (f_1 \rightarrow n'', f_2 \rightarrow n'') \succeq (f_1 \rightarrow n', f_2 \rightarrow n'') \succeq (f_1 \rightarrow n'', f_1 \rightarrow n') \quad (7.12)$$

based on the following information:

$$\mathcal{U}_c(\cdot) |_{(f_1 \rightarrow n', f_2 \rightarrow n')} \succeq \mathcal{U}_c(\cdot) |_{(f_1 \rightarrow n'', f_2 \rightarrow n'')} \succeq \mathcal{U}_c(\cdot) |_{(f_1 \rightarrow n', f_2 \rightarrow n'')} \succeq \mathcal{U}_c(\cdot) |_{(f_1 \rightarrow n'', f_1 \rightarrow n')} \quad (7.13)$$

On the other hand, each switch  $s$  decides the amount of bandwidth to be allocated for each flow  $f_n \in \mathcal{F}_s(t)$ . Using Karush-Kuhn-Tucker (KKT) condition, we get:

$$\begin{aligned} \mathcal{L} = \mathcal{U}_s(t) - \sum_{f_n \in \mathcal{F}_s(t)} \lambda_{1,n}(r_n(t) - r_n^{min}) + \sum_{f_n \in \mathcal{F}_s(t)} \lambda_{2,n}(r_n(t) - r_n^{max}) \\ - \lambda_3 [C_s - \sum_{f_n \in \mathcal{F}_s(t)} r_n(t) - \sum_{f_d \in \mathcal{F}_s(t)} r_d(t)] \quad (7.14) \end{aligned}$$

where  $\lambda_{1,n}$ ,  $\lambda_{2,n}$ , where  $f_n \in \mathcal{F}_s(t)$ , and  $\lambda_3$  are Lagrangian multipliers. Additionally, we have:

$$\lambda_{1,n}, \lambda_{2,n}, \lambda_3 \geq 0, \quad \forall f_n \in \mathcal{F}_s(t) \quad (7.15)$$

$$\left. \begin{aligned} \lambda_{1,n}(r_n(t) - r_n^{min}) &= 0 \\ \lambda_{2,n}(r_n(t) - r_n^{max}) &= 0 \end{aligned} \right\}, \quad \forall f_n \in \mathcal{F}_s(t) \quad (7.16)$$

$$\lambda_3 \left( C_s - \sum_{f_n \in \mathcal{F}_s(t)} r_n(t) - \sum_{f_d \in \mathcal{F}_s(t)} r_d(t) \right) = 0 \quad (7.17)$$

Hence, performing  $\nabla_{r_n(t)} \mathcal{L} = 0$ , we get:

$$r_n^*(t) = \left[ \frac{M_n}{\phi_s} \right]^{\frac{1}{2}} \left( \frac{\zeta_s}{r_n^{max}} + \frac{\varphi_s}{C_s} \right)^{-\frac{1}{2}} \quad (7.18)$$

Additionally, we get that  $\nabla_{r_n(t)}^2 \mathcal{L} \leq 0$ . Hence, we get that at  $r_n(t) = r_n^*(t)$ ,  $\mathcal{U}_s(\cdot)$  is maximized.

### 7.3 Proposed Algorithms

In D2M, initially the controller decides the  $x_{ns} \in \{0, 1\}$ ,  $\forall f_n \in \mathcal{F}(t)$ , and tries to minimize overall network delay by using Algorithm 7.1. We assume that the controller has the knowledge that the IoT devices are connected to which access points (APs) and the corresponding SDN switches. Moreover, we consider that the controller knows the set of IoT devices which are interested in downloading the data. On the other hand, using Algorithm 7.2, each switch  $s$  decides the amount of bandwidth to be allocated to the associated flows  $\mathcal{F}_s(t)$ , while satisfying the constraints mentioned in Equations (7.2) and (7.3). Algorithm 7.2 needs to be executed by each switch, individually and non-cooperatively. The time complexity for Algorithm 7.1 is  $O(\max(|\mathcal{F}|^{|S|}, |S|^{|\mathcal{F}|}))$ . On the other hand, the time complexity for Algorithm 7.2 is  $O(K)$ , where  $K \in \mathbb{R}^+$ . Therefore,



## 7.4. Performance Evaluation

---

the over complexity of D2M is  $O(\max(|\mathcal{F}|^{|\mathcal{S}|}, |\mathcal{S}|^{|\mathcal{F}|}) + K) \approx O(\max(|\mathcal{F}|^{|\mathcal{S}|}, |\mathcal{S}|^{|\mathcal{F}|}))$ .

---

### Algorithm 7.1 Optimal Flow Association Vector

---

**INPUTS:**

- 1:  $\mathcal{A}, \bar{\mathcal{A}}(t), \mathcal{F}(t), \bar{\mathcal{F}}(t), \mathcal{S}$
- 2:  $E_n^{res}(t), E_{max}, h_{f_n}(t), a_n(t), \forall f_n \in \mathcal{F}$
- 3:  $\alpha_n, \beta_n, \gamma_n$

**OUTPUT:**

- 1:  $\{x_{ns}\}, \forall f_n \in \mathcal{F}$

**PROCEDURE:**

- 1: Find Cartesian product of  $\mathcal{F}$  and  $\mathcal{S}$
  - 2: **do**
  - 3:     Chose a vector of  $\{f_n, s | \forall f_n \in \mathcal{F}\}$
  - 4:     Calculate  $\mathcal{U}_n(\cdot)$  using Equation (7.7);
  - 5: **while** each element in Cartesian set is not visited for  $|\mathcal{S}|$  times;
  - 6: Calculate a preference relation among the calculated  $\mathcal{U}_n(\cdot)$  values
  - 7: Select the vector with highest preference value
  - 8: **return** Corresponding  $\{x_{ns}\}, \forall f_n \in \mathcal{F}$
- 

**Table 7.1:** Simulation Parameters

Parameter	Value
Simulation Area	1000 $m \times$ 1000 $m$
Number of Nodes	1000-50000
Number of Switches	4
Number of Servers	3
Velocity of Source Nodes	5 $m/s$
Capacity of Switches	10 $Gbps$
Data chunks generated	1000
Size of each data chunk	800 $Mb$
Mobility model (MM)	Random Gauss-Markov

## 7.4 Performance Evaluation

In this section, we analyze the performance of D2M through simulation by varying the number of heterogeneous IoT devices in terms of the bandwidth capacity.

---

**Algorithm 7.2** Optimal Data-rate for Each Flow

---

**INPUTS:**

- 1:  $\mathcal{F}_s(t)$ ,  $C_s$ ;  $r_d(t)$ ,  $\forall f_d \in \mathcal{F}_s(t)$ ;  $r_n^{min}$ ,  $r_n^{max}$ ,  $M_n$ ,  $\forall f_n \in \mathcal{F}_s(t)$
- 2:  $\zeta_s$ ,  $\phi_s$ ,  $\varphi_s$
- 3:  $\delta$

▷ Increment factor

**OUTPUT:**

- 1:  $\{r_n^*(t)\}$ ,  $\forall f_n \in \mathcal{F}_s(t)$

**PROCEDURE:**

- 1: **for** each  $f_n \in \mathcal{F}_s(t)$  **do**
  - 2:      $r_n^*(t) \leftarrow r_n^{min}$
  - 3: **end for**
  - 4: Calculate  $\mathcal{U}_s(\cdot)$  using Equation (7.9)
  - 5: **do**
  - 6:     **for** each  $f_n \in \mathcal{F}_s(t)$  **do**
  - 7:          $r_n(t) \leftarrow r_n^* + \delta$
  - 8:         **if then**  $r_n(t) < r_n^{max}$
  - 9:              $\mathcal{U}_s^{prev}(\cdot) \leftarrow \mathcal{U}_s(\cdot)$
  - 10:             Calculate  $\mathcal{U}_s(\cdot)$  using Equation (7.9)
  - 11:             **if then**  $\mathcal{U}_s(\cdot) \geq \mathcal{U}_s^{prev}(\cdot)$
  - 12:                  $r_n^*(t) \leftarrow r_n(t)$
  - 13:             **end if**
  - 14:         **end if**
  - 15:     **end for**
  - 16: **while** There is any change in  $\{r_n^*(t)\}$ ,  $\forall f_n \in \mathcal{F}_s(t)$
  - 17: **return**  $\{r_n^*(t)\}$ ,  $\forall f_n \in \mathcal{F}_s(t)$
-

## 7.4. Performance Evaluation

---

### 7.4.1 Simulation Parameters

For the performance evaluation, we simulated using JAVA-based platform and deployed the IoT devices randomly over a terrain of  $1000 \times 1000 m^2$ . However, the switches and the routers are deployed in a grid fashion, while ensuring full coverage. We considered that the source IoT devices generate 1000 number of data chunks, and the size of each data chunk is 800 Mb, as mentioned in Table 7.1. Motivated by the device distribution of the Internet [13], we considered that the distribution of IoT device capacities follows the distribution mentioned in Table 7.2 [103].

**Table 7.2:** Node Capacity Distribution

Capacity (Kbps)	Nodes (%)
128	20
384	40
1000	25
5000	15

### 7.4.2 Benchmarks

The performance of D2M is evaluated while comparing with two existing schemes for DCNs — the Lock-Step Broadcast Tree based big-data broadcasting (LSBT) [13] and the Multicast Fat-Tree Data Center Networks (BLO) [21] schemes. In LSBT, Wu *et al.* [13] proposed a big-data broadcasting scheme, while forming a *Lock Step Broadcast Tree*. The authors also considered that the source device, which has the maximum capacity in the network, is at the root of the tree. On the other hand, in BLO [21], Guo and Yang proposed a fat-tree based DCN. In BLO, the authors tried to minimize the number of core switches needed to overcome the problem of over subscriptions. Additionally, the authors overlooked the problem of balanced bandwidth distribution. Moreover, these works do not consider SD-DCN in the presence of mobile IoT devices. In D2M, we improve the network performance for data multicasting, while ensuring optimal throughput and delay of the network.

### 7.4.3 Performance Metrics

We evaluate the performance of D2M using the following metrics:

**Network Delay:** We define network delay as the total time required to complete the data multicasting in SD-DCN.

**Hop-Count for Network Flows:** We calculate the overall hop-count for the data multicasting. With the increase in hop-count, the real-timeliness of the multicasted data degrades.

**Network Throughput:** The network throughput signifies that the amount of data successfully transmitted from the source node to the destination. With efficient load balancing, network throughput increases significantly.

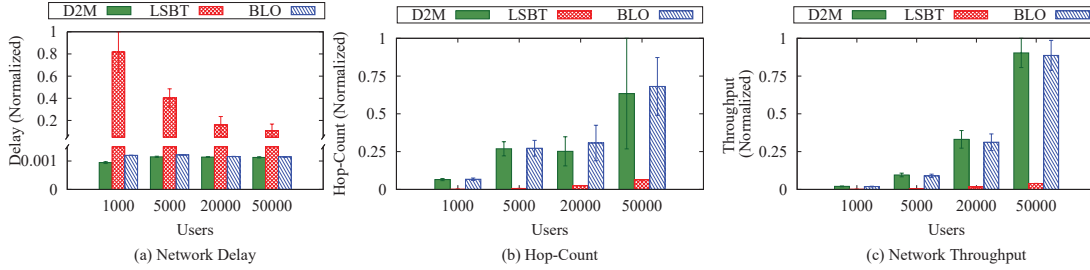


Figure 7.2: Performance Analysis of D2M

### 7.4.4 Results and Discussions

Figure 7.2(a) depicts that using D2M, the network delay decreases by 21.32% and 99.29% than using BLO and LSBT, respectively. In LSBT, delay increases significantly due to the fact that LSBT multicasts data in intra-networks. Hence, only after it reaches to a server at the edge-tier, the data gets multicasted in the inter-network. On the other hand, the network delay using BLO is higher than D2M due to inefficient network load balancing. Similarly, from Figure 7.2(b), we observed that the hop-count using LSBT is

## 7.5. Concluding Remarks

---

significantly low as the source IoT device multicasts data in the intra-network. Therefore, in fat-tree architecture, LSBT cannot ensure data transfer to each IoT device. However, we yield that using D2M, the hop-count reduces by 3.69-7.44% than using BLO. We observe that D2M ensures efficient load balancing in SD-DCN.

On the other hand, from Figure 7.2(c), we observed that using D2M, the network throughput increases by 6.13% and 95.32% than using BLO and LSBT, respectively. As mentioned earlier, using LSBT, the source node can only multicast data in intra-network. Therefore, network throughput reduces significantly. The network throughput using D2M is comparable with the throughput achieved using BLO, as both schemes are designed for DCN with fat-tree architecture. However, as D2M is designed for SD-DCN, the controller has an overview of the network, which BLO lacks in. Thereby, we observe that D2M ensures high utilization of network capacity while distributing the network load efficiently compared to the existing schemes — LSBT and BLO.

## 7.5 Concluding Remarks

In this Chapter, we formulated a single leader multiple follower Stackelberg game-based D2M scheme to ensure high utilization of network capacity and efficient load balancing in SD-DCN. We observed that D2M ensures the reduction in network delay in the presence of mobile IoT devices. Additionally, using D2M, network throughput increases. From the simulation, we observed that D2M outperforms the other existing schemes — LSBT and BLO. In particular, we observed that using D2M, the network throughput increased by 6.13-95.32% than using existing schemes, while ensuring 21.32-99.29% reduction in delay.

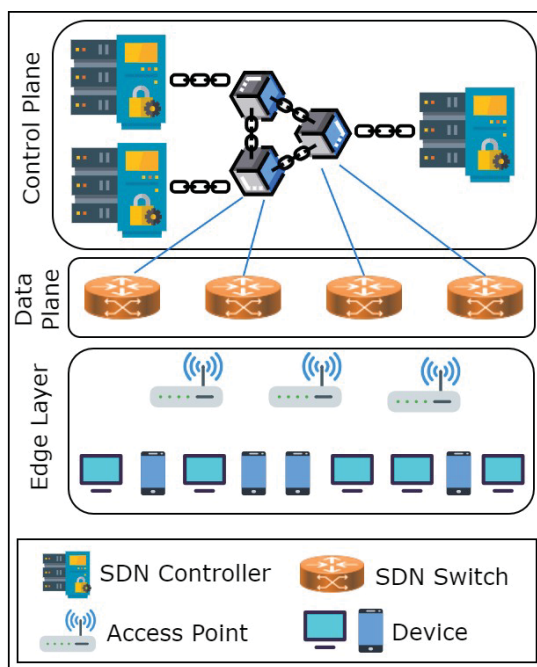


## Chapter 8

# Multi-Tenant Flow-Table Partitioning

In this Chapter, we introduce a utility game theory-based flow-table partitioning scheme, named BIND, for maximizing the network sustainability and minimizing the network overhead and delay in the distributed multi-tenant SDN. We use utility game to decide the flow-rules to be replaced while ensuring fairness among the controllers. In order to ensure fairness among the multi-tenant controllers, we use a blockchain-based flow-table partitioning. In BIND, we consider that the flow-tables are virtually owned by each controller. Hence, to replace any flow-rule, the controllers need to decide the flow-rule unanimously, in which blockchain plays a key role. In BIND, on receiving a Packet-In message, the controller checks for the free flow-table space, and installs the flow-rule if ternary content-addressable memory (TCAM) is available. Otherwise, it requests other controllers to elect a subset of flow-rules which are eligible to be replaced. Thereafter, the controller selects a flow-rule having the highest payoff in the proposed utility game-based scheme. In BIND, we also introduce the *flow-priority*, which helps in ensuring network sustainability.

This Chapter is organized as follows. The system architecture considered in BIND



**Figure 8.1:** Schematic Diagram of Multi-Tenant SDN

scheme is discussed in Section 8.1, while mentioning the assumptions. Section 8.2 focuses on the formulation of BIND using single-leader-multiple-follower Stackelberg game. The algorithms proposed for BIND scheme are discussed in Section 8.3. We evaluate the performance of BIND in Section 8.4 while comparing with the existing schemes. Finally, Section 8.5 concludes this Chapter.

## 8.1 System Model

In this Chapter, we consider a distributed multi-tenant SDN comprising of multiple SDN-controllers and multiple SDN-switches. Each switch  $s \in \mathcal{S}$ , where  $\mathcal{S}$  represents a set of SDN switches, is connected with a set  $\mathcal{C}$  of SDN controllers, as shown in Figure 8.1. Each switch  $s \in \mathcal{S}$  has a flow-rule capacity of  $\rho_s$ . The controllers are not energy constrained and have access to the  $\mathcal{S}$  set of switches, and can change the flow-rules of the shared flow-tables as per requirement. The controllers use *permissioned blockchain*



## 8.1. System Model

---

*network* [107] to record and verify the changes in the flow-tables. We consider that, after receiving a Packet-In message, the concerned controller initiates a transaction having flow replacement information, while considering that the Packet-In messages are not correlated. The transactions are digitally signed. The controller generates a block for each transaction and adds it to the blockchain. We consider that each IoT device  $n \in \mathcal{A}_c$  registers to controller  $c \in \mathcal{C}$  for data transmission, where  $\mathcal{A}_c$  denotes a set of IoT devices connected with controller  $c$ . Within a duration of  $\Delta$ , each IoT device  $n$  generates  $F_n(\Delta)$  set of flows and each controller  $c$  receives  $F_c(\Delta)$  set of Packet-In messages. Therefore, we get that:

$$F_c(\Delta) \subseteq \bigcup_{n \in \mathcal{A}_c} F_n(\Delta) \quad (8.1)$$

Additionally, we consider that each controller  $c$  sets the priority for each flow-rule based on the type, i.e., mice or elephant flow, and the content of the flow<sup>1</sup>, where  $\mathbb{P}$  represents the set of priorities of the flows. The switches have limited TCAM space, hence, in a dynamic network, the flow-rules may need to be replaced frequently. However, the priority  $\eta_f$  of each flow  $f \in \bigcup_c F_c(\cdot)$  ensures the sustainability of the network, as defined in Definition 8.1. We consider a flow to be an active flow if it has transmitted data within  $\delta$  duration.

**Definition 8.1.** *We define the sustainability of the network,  $\zeta$ , as the percentage of flows not blocked by the flows having low priority. In other words, the sustainability of the network varies proportionally with the number active flow with high priority which are not interrupted by any flow with low priority. Mathematically,*

$$\zeta = 1 - \frac{\left| \left\{ f_i \in \bigcup_c F_c(\cdot) \mid (x_{f_i} = 1, x_{f_j} = 0, \eta_{f_i} < \eta_{f_j}) \right\} \right|}{\left| \left\{ f_i \in \bigcup_c F_c(\cdot) \right\} \right|} \quad (8.2)$$

---

<sup>1</sup>We consider that the controller identifies the content of the flow based on the received meta-data along with Packet-In message.

where  $x_f$  is binary variable and denotes the presence of the flow-rule in flow-table corresponding to active flow  $f$ . Therefore, we have:

$$x_f = \begin{cases} 1, & \text{if rule for active flow } f \text{ is installed in flow-table} \\ 0, & \text{otherwise} \end{cases} \quad (8.3)$$

**Problem Scenario** We consider a distributed multi-tenant SDN in the absence of a centralized controller. Multiple controllers share the same flow-space among themselves. We consider that the controller follows the *soft* flow-table partitioning in the proposed SDN. Hence, it may lead to a monopoly situation, where a subset of controllers handle a huge number of flows and modify the flow-rules accordingly. To avoid such a scenario, in this Chapter, we aim to propose a blockchain-based flow-table partitioning scheme for distributed multi-tenant SDNs in the absence of a centralized SDN-controller.

**Assumptions** To design the scheme for blockchain-based flow partitioning in distributed multi-tenant SDNs, we assume that:

1. There is no centralized SDN-controller to coordinate in multi-tenant SDN.
2. The SDN-controllers do not misbehave and cooperate.
3. The controllers are capable of ensuring the validity of the metadata of Packet-In messages.

## 8.2 BIND: The Proposed Blockchain-Based Flow-Table Partitioning Scheme

To design the interactions among the SDN-switches at the data plane and the SDN-controllers at the control panel, we use a utility game [108, 109]. This is a cooperative game, where the SDN-controllers act cooperatively in sharing the space of the flow-tables while ensuring the sustainability of the network, i.e., the high priority flows are

## 8.2. BIND: The Proposed Blockchain-Based Flow-Table Partitioning Scheme

---

not blocked by the low priority flows. Therefore, in the proposed scheme, BIND, the controllers are the players. On the other hand, the flow-space or the flow-tables are considered to be resources, which need to be shared among the player optimally, while ensuring the sustainability of the network. Moreover, the controllers need to ensure that the optimal network throughput is achieved.

### 8.2.1 Justification for Using Utility Game

For efficient flow-table partitioning, we need to ensure that the monopoly is not present among the controllers. Additionally, the controllers need to ensure network sustainability with optimal throughput. Due to the absence of any centralized coordinator, the controllers decide a subset of flow-rules which can be replaced. Thereafter, they decide which flow-rule to be replaced using cooperation. The utility game plays an important role in the aforementioned process. The actions of the controllers get registered in the blockchain, whenever there is a change in the flow-table. To generate the payoff values of the utility functions, the controllers refer to the previous blocks in the blockchain. Thereby, we infer that the utility game along with the blockchain among the controllers ensures that the controllers get a fair chance to update the flow-table. Additionally, high network sustainability is ensured.

### 8.2.2 Game formulation

To achieve optimal flow-table partitioning, we use utility game. In the proposed blockchain-based flow-table partitioning scheme, named BIND, the controllers act as the players and choose strategies, i.e., flow-rule replacement, for ensuring network sustainability, while ensuring high network throughput. The controllers use the blockchain to record the flow-rule updates as blocks in the multi-tenant SDN. In the absence of a centralized coordinator, the controllers use cooperation to ensure the fair chance in performing the flow-rule update. Thereby, using blockchain and utility game theory, the proposed

scheme, BIND, ensures to avoid monopoly among the controllers, which is a significant drawback of the soft flow-table partitioning. In BIND, each block in the blockchain encapsulates the following information:

1. *Controller-Specific Flow-Rule Replacement Counter*: Each controller  $c \in \mathcal{C}$  increments the flow-rule replacement counter,  $\mathbb{C}_c$ , and append the information in the corresponding block.
2. *Total Number of Flow-Rule Replacement*: It represents the total number of flow-rule replacement  $\mathbb{R}$  occurred after the last network update. This counter is considered to be a *global* counter.
3. *Tolerable Waiting Time*: The controllers update the tolerable waiting time  $\mathbb{T}_f$  for each flow-rule  $f$ , when there is a change in the flow-table. The controllers follow an  $O(K)$  Markov predictor to update  $\mathbb{T}_f, \forall f$  [110], where  $K$  is constant.
4. *Elapsed Time*: The controllers update the elapsed time after recent use,  $t_f$ , for each flow  $f$ , which signifies the information of the least recently used flow-rules.
5. *Flow-Rule Priority*: The priority  $\eta_f$  of the newly installed flow-rule  $f$  is considered to be the same as the installed flow-rule priority. The controllers refer to this parameter to identify the subset of flow-rules that are eligible for replacement in the flow-tables.

### 8.2.3 Replacement Eligibility Factor for Each Flow-Rules

Each controller  $c$  calculates the payoff of each flow-rule  $f \in F_c(\cdot)$  while considering the two parameters — tolerable waiting time  $\mathbb{T}_f$  and elapsed time  $t_f$ . Based on these parameters, the controllers calculates the *eligibility* of the flow-rules to be replaced. We define the *replacement-eligibility factor*  $\mathbb{E}_f$  for each flow-rule  $f$  in terms of probability, as mentioned in Definition 8.2.

## 8.2. BIND: The Proposed Blockchain-Based Flow-Table Partitioning Scheme

---

**Definition 8.2.** *Replacement-eligibility factor  $\mathbb{E}_f$  of each flow-rule  $f$  is defined as the ratio of the predicted duration before receiving the next packet for flow-rule  $f$  and tolerable waiting time (predicted). Therefore, we get:*

$$\mathbb{E}_f = \frac{\mathbb{T}_f - t_f}{\mathbb{T}_f} \quad (8.4)$$

Therefore, we get that in BIND, the flow-rule  $f$  with high replacement-eligibility factor has higher probability to get replaced by the flow-rule associated with the new incoming flow  $f_n$ , while considering the following constraint is satisfied:

$$\eta_f < \eta_{f_n} \quad (8.5)$$

where  $\eta_f$  and  $\eta_{f_n}$  denote the priorities of the flow-rules  $f$  and  $f_n$ , respectively. Each controller elects a flow-rule having maximum  $\mathbb{E}_f$  value which may be replaced.

### 8.2.4 Utility Function of Each Controller

Based on the elected flow-rule, each controller  $c$  calculates the payoff of utility function  $\mathcal{U}_c(\cdot)$ . The utility function  $\mathcal{U}_c(\cdot)$  signifies the probability of the flow-rule to be replaced associated with controller  $c$ . In order to design the utility function  $\mathcal{U}_c(\cdot)$ , the controllers consider the parameters such as flow-rule replacement counter  $\mathbb{C}_c$ , the total number of flow-rule replacement  $\mathbb{R}$ , and replacement-eligibility factor  $\mathbb{E}_f$  of the elected flow  $f$ . Additionally, the controllers consider the parameter — change in throughput  $\phi_f$  — by replacing the flow-rule  $f$ . The utility function  $\mathcal{U}_c(\cdot)$  needs to satisfy the following properties:

1. Flow-rules associated with each controller are treated with fairness and monopoly does not occur. Therefore, we consider that:

$$\frac{\partial \mathcal{U}_c(\cdot)}{\partial \mathbb{C}_c} < 0 \quad (8.6)$$

2. We aim to reduce the number of replacement in the flow-tables. Therefore, we consider that utility function  $\mathcal{U}_c(\cdot)$  needs to follows the following constraint:

$$\frac{\partial \mathcal{U}_c(\cdot)}{\partial \mathbb{E}_f} > 0 \tag{8.7}$$

3. We aim to increase the overall network throughput for ensuring high bandwidth utilization. Therefore, we consider that  $\mathcal{U}_c(\cdot)$  has high utility value if the change in the throughput is high.

$$\frac{\partial \mathcal{U}_c(\cdot)}{\partial \mathbb{D}_f} > 0 \tag{8.8}$$

where  $\mathbb{D}_f$  denotes the change in throughput for replacing flow-rule  $f$ .

Therefore, we define the utility function  $\mathcal{U}_c(\cdot)$  as follows:

$$\mathcal{U}_c(\cdot) = \mathbb{E}_f \mathbb{D}_f \left( 1 - \frac{\mathbb{C}_c}{\mathbb{R}} \right) \tag{8.9}$$

where  $f \in F_c$ . The controllers aim to maximize the payoff value of the utility function  $\mathcal{U}_c(\cdot)$  to ensure network sustainability and throughput.

### 8.3 Proposed Algorithms

If flow-rule space is available in the flow-table, the controller that received the Packet-In message installs the flow-rule and generates a block, accordingly. However, in case of no flow-rule space availability, the controllers act cooperatively and decide the flow-rule to be replaced while ensuring the enhanced performance of the network with high network sustainability. To achieve the aforementioned goal in BIND, we propose two algorithms — Flow-Rule Election (FLE) and Flow-Rule Replacement (FRR). In BIND, once a Packet-In message is received by a controller, each controller executes the FLE

### 8.3. Proposed Algorithms

---

---

**Algorithm 8.1** FLE: Flow-Rule Election in BIND

---

**INPUTS:**

- 1:  $F_c$  ▷ Set of flow-rules maintained by controller  $c$
- 2:  $f_n$  ▷ New flow-rule for received Packet-In message
- 3:  $\eta_f, t_f, \mathbb{T}_f \forall f \in F_c$
- 4:  $\eta_{f_n}$

**OUTPUTS:**

- 1:  $f_c \in F_c$  ▷ Elected flow-rule by controller  $c$
- 2:  $\mathbb{E}_{f_c}$  ▷ Replacement eligibility factor of elected flow-rule

**PROCEDURE:**

- 1:  $\mathcal{E}_c \leftarrow \{\emptyset\}$
  - 2: **for** Each  $f \in F_c$  **do**
  - 3:     **if**  $\eta_f < \eta_{f_n}$  **then**
  - 4:         Calculate  $\mathbb{E}_f$  using Equation (8.4)
  - 5:          $\mathcal{E}_c \leftarrow \mathcal{E}_c \cup \mathbb{E}_f$
  - 6:     **end if**
  - 7: **end for**
  - 8: **if**  $\mathcal{E}_c \neq \{\emptyset\}$  **then**
  - 9:     Select flow-rule  $f_c$  such that  $\mathbb{E}_{f_c} \geq \mathbb{E}_f$  where  $f \neq f_c$  and  $f, f_c \in F_c$
  - 10:     **for** Each  $f \in F_c / \{f_c\}$  **do**
  - 11:         Update  $\mathbb{T}_f$  using Markov predictor [110]
  - 12:     **end for**
  - 13:     **return**  $\{f_c, \mathbb{E}_{f_c}\}$
  - 14: **else**
  - 15:     **return**  $\{NULL, NULL\}$
  - 16: **end if**
-

---

**Algorithm 8.2** FRR: Flow-Rule Replacement in BIND

---

**INPUTS:**

- |    |  |   |
|----|--|---|
| 1: | $f_c, \mathbb{E}_{f_c}, \forall c \in \mathcal{C}$ | ▷ Outputs from FLE Algorithm                        |
| 2: | $\mathbb{C}_c, \forall c \in \mathcal{C}$          | ▷ Controller-specific flow-rule replacement counter |
| 3: | $d_{f_c} \forall c \in \mathcal{C}$                | ▷ Throughput of the selected flow-rules             |
| 4: | $f_n$  | ▷ New flow-rule for received Packet-In message      |
| 5: | $d_{f_c}$  | ▷ Throughput of the new flow-rule $f_n$             |
| 6: | $\mathbb{R}$                                       | ▷ Total number of flow-replacement                  |

**OUTPUTS:**

- |    |           |  |
|----|-----------|--|
| 1: | $x_{f_n}$ | ▷ Presence of new flow-rule in the flow-tables |
| 2: | $f^*$     | ▷ Flow-rule replaced by flow-rule $f_n$        |

**PROCEDURE:**

- 1:  $V \leftarrow \{\emptyset\}$
  - 2: **for** Each  $c \in \mathcal{C}$  **do**
  - 3: **if**  $f_c \neq NULL$  **then**
  - 4:  $\mathbb{D}_{f_c} \leftarrow d_{f_n} - d_{f_c}$
  - 5: Calculate  $\mathcal{U}_c(\cdot)$  using Equation (8.9)
  - 6:  $V \leftarrow V \cup \mathcal{U}_c(\cdot)$
  - 7: **end if**
  - 8: **end for**
  - 9: **if**  $V == \{\emptyset\}$  **then**
  - 10: **return**  $\{0, NULL\}$
  - 11: **else**
  - 12:  $f^* \leftarrow f_c$  such that  $\mathcal{U}_c(\cdot) \geq \mathcal{U}_{c'}(\cdot)$  where  $c \neq c'$  and  $c, c' \in \mathcal{C}$
  - 13: **return**  $\{1, f^*\}$
  - 14: **end if**
-



### 8.3. Proposed Algorithms

---

algorithm (Algorithm 8.1), distributively. Using the FLE algorithm, the controllers in the multi-tenant SDN elect a subset of flow-rules which are eligible to be replaced. Initially, using the FLE algorithm, each controller selects a subset of flow-rules based on flow-priority, i.e., satisfy the constraint in Equation (8.5), (refer to Line 2). Using Algorithm 8.1, each controller aims to elect *at most one* flow-rule having maximum replacement eligibility factor.

The controller that receives the Packet-In message, executes the FRR algorithm (Algorithm 8.2) and decides if the new flow-rule is to be installed or discarded. Additionally, using the FRR algorithm, the controller decides which flow-rule is to be replaced to install the new flow-rule. Using the FRR algorithm (Lines 4-5), the controller calculates the utility function, i.e., the probability of elected flow-rules to be replaced by the new flow-rule. Accordingly, the controller generates a block and adds it to the blockchain.

**Complexity Analysis** In BIND, we take advantage of the distributed architecture of blockchain for designing a scheme for flow-table partitioning in the multi-tenant SDN. As mentioned earlier, the proposed scheme, BIND, has two components — flow-rule election and flow-rule replacement. We observe that the time complexity of Lines 2–5 and 10–12 (Algorithm 8.1) is  $O(|F_c|)$ . Therefore, we get that the time complexity for the FLE algorithm in BIND  $O(|F_c|)$  for controller  $c$ . Accordingly, we get that, in BIND, the time complexity of the FLE algorithm for the overall network is  $O(\max |F_c|)$ . On the other hand, we observe that the time complexity of Lines 2-8 (Algorithm 8.2) is  $O(|\mathcal{C}|)$ . Therefore, the time complexity of the FRR algorithm is  $O(|\mathcal{C}|)$ . Hence, the overall complexity of the proposed scheme, BIND, is  $O(|\mathcal{C}| + \max |F_c|)$ .

Moreover, the space complexity of the FLE algorithm in BIND is  $O(|\max F_c|)$ , where the space complexity for each controller  $c$  is  $O(|F_c|)$ . On the other hand, the space complexity for the FRR algorithm in BIND is  $O(|\mathcal{C}|)$ . Therefore, similar to the time complexity, we observe that the space complexity of the proposed scheme, BIND, is  $O(|\mathcal{C}| + \max |F_c|)$ .

## 8.4 Performance Evaluation

In this section, we analyze the performance of the proposed scheme, BIND, for varying number of flows in multi-tenant SDN. Generic test-bed information for BIND is provided in Table 8.1.

**Table 8.1:** System Specification

Parameter	Value
Processor	Intel(R) Core(TM) i5-2500 CPU @ 3.30 GHz
RAM	4 GB DDR3
Disk Space	500 GB
Operating System	Ubuntu 16.04 LTS

### 8.4.1 Simulation Parameters

For simulation, we considered the number of SDN switches and the controllers to be 20 and 5, respectively. We varied the number of flows as mentioned in Table 8.2. We consider that each flow generates data traffic at the rate of 0.2 million packets per second (*mpps*).

**Table 8.2:** Simulation Parameters

Parameter	Value
Simulation area	1000 $m \times 1000 m$
Number of controllers	5
Number of switches	20
Number of flows	1000, 2000, 3000
Flow priorities	1-5
Data-rate requirement per flow	50-100 kbps
Maximum data-rate per switch	$10^3$ kbps

### 8.4.2 Benchmarks

The performance of the proposed scheme, BIND, is evaluated by comparing with the two schemes — hard flow-table partitioning (HARD) and soft flow-table partitioning

## 8.4. Performance Evaluation

---

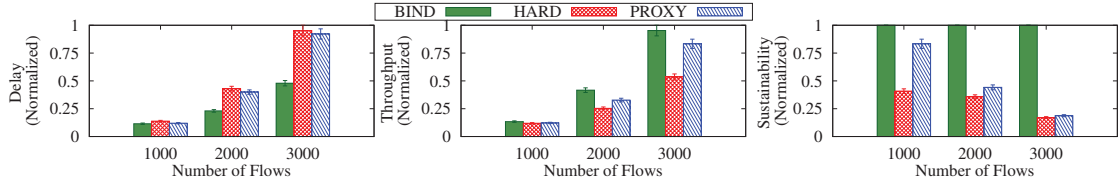


Figure 8.2: Comparison of BIND with Other Schemes

in the presence of proxy controller (PROXY). In HARD, we consider that controllers have non-overlapping and restricted flow-table access. On the other hand, in PROXY, we consider that the controllers use soft flow-table partitioning in the presence of proxy controller.

We evaluated the performance of the proposed scheme, BIND, using the metrics such as network delay, network throughput and network sustainability, as discussed below:

### 8.4.3 Performance Metrics

We have evaluated the performance of BIND using the following metrics:

**Network Delay:** The overall network delay is the composite of the flow setup delay and the data traffic delay. However, in BIND, we focus on reducing only the flow setup delay. Hence, by network delay, we consider the flow setup delay.

**Network Throughput:** The network throughput signifies that the amount of data successfully transmitted from the source node to the destination. With efficient rule placement in multi-tenant SD-DCN, network throughput increases significantly.

**Network Sustainability:** As mentioned earlier, the network sustainability is defined as the ratio of the number active flow with high priority which are not interrupted by any flow with low priority and the total number of active flows.

### 8.4.4 Results and discussions

From Figure 8.2(a), we observe that with the increase in the number of flows, the flow setup delay reduces by 48.1-49.7% using BIND, due to the distributed nature. However, in HARD and PROXY, the flow setup is a centralized approach. Hence, we observe that, in HARD and PROXY, the delay increases in polynomial curve, whereas the delay using BIND increases linearly.

We observe that using BIND, the achieved throughput is higher by 12.5-14.3% as compared to HARD and PROXY, as shown in Figure 8.2(b). We observed that, in HARD and PROXY, the flow-rule gets replaced based on the time-stamp only, however in BIND, we ensure that the flow-rules having high throughput get priority in flow-rule replacement.

From Figure 8.2(c), we observe that BIND ensures 100% network sustainability, whereas using HARD and PROXY, the network sustainability decreases with the increase in the number of flows. In BIND, while evaluating the eligibility factor of the flow-rules, we ensure that the high priority active flows are not replaced by the low priority flows, which results in high network sustainability.

## 8.5 Concluding Remarks

In this Chapter, we proposed BIND, a blockchain-based flow-table partitioning scheme, for distributed multi-tenant SDN. Using blockchain, we ensured that the controllers are synchronized and cooperative in nature. In BIND, we used utility game to propose the distributed algorithm for flow-rule election, where each controller distributively identifies the flow-rules' replacement eligibility factors, and elects a single flow-rule for replacement. Thereafter, we considered a utility game-based centralized algorithm for flow-rule replacement to be performed by the controller receiving the Packet-In message. We observed that BIND ensures fairness in flow-rule replacement for the controllers in

## 8.5. Concluding Remarks

---

a distributed multi-tenant SDN. Through simulation, we observed that the flow setup delay increases linearly using BIND. BIND also ensures high throughput and 100% network sustainability, thereby outperforming the benchmark schemes – HARD and PROXY. In particular, we observed that using BIND, the flow setup delay reduces by 48.1-49.7%, while ensuring 12.5-14.3% increase in network throughput than using the existing schemes.



## Chapter 9

# Conclusion

The objective of this Thesis is to design different data traffic management schemes suitable for SD-DCN in the presence of heterogeneous elephant and mice flows. We considered various challenges such as the presence of heterogeneous flows, performance limitations and corresponding optimal values, limited TCAM space, and limited network bandwidth. Chapters 3-4 presented the analytical schemes proposed in this Thesis. On the other hand, Chapters 5-7 focused on designing data traffic management schemes for SD-DCN in the presence of heterogeneous flows. Further, Chapter 8 discussed flow-table partitioning for distributed multi-tenant SD-DCN in the presence of heterogeneous IoT flows.

In this final Chapter, we take a holistic view of our work discussed so far. We discuss the summary of this Thesis in Section 9.1. We list out the major contributions of the Thesis in Section 9.2. In Section 9.3, we discuss the limitations of our work. Finally, in Section 9.4, we conclude the Thesis while citing future directions.

### 9.1 Summary of the Thesis

This Thesis was presented in nine chapters. Chapter 1 presented a brief introduction to SD-DCN, and discussed the motivation of the work while citing the main objectives of

this Thesis.

In Chapter 2, we surveyed the existing literature on theoretical analysis of SDN, and resource management in SDN and DCN. Finally, we summarized the problem area based on the limitations of the existing schemes.

Chapter 3 analyzed the performance of packet flow through an OpenFlow switch in SD-DCNs and proposed an analytical model, named AMOPE, to define the probabilistic bounds of the performance metrics of the OpenFlow switch. We modeled the packet flow steps in an OpenFlow switch using Markov chain and calculated the theoretical probabilities of the packet to be in different states. Additionally, we calculated the probabilities of a packet being at output action state, getting dropped, and getting forwarded to the controller, theoretically. We also verified the theoretical findings using the MATLAB simulation platform. Through simulations, we observed that approximately 60% of the processed packets are sent to output action, 31% of the processed packets are sent to the controller, and the remaining processed packets are dropped in an OpenFlow switch.

Chapter 4 presented the scheme for analyzing the optimum buffer size of an OpenFlow switch in order to ensure QoS in OpenFlow systems. We analyzed the optimum packet arrival and processing rates, and the average waiting time of packets. In OPUS, we modeled the architecture of an OpenFlow switch as a I-M/M/1/K queue, while considering that there are  $I$  ingress buffers. Each buffer has  $K$  memory blocks in an OpenFlow switch. We analyzed the optimum number of buffers with the optimum value of memory of each buffer. Additionally, we evaluated the optimum packet arrival and processing rates of an OpenFlow switch using OPUS. Simulation-based analysis exhibited that with two times increase in packet processing rate, the packet arrival rate can be increased by 26.15-30.4%. We inferred that for an OpenFlow system, the minimum buffer size is 0.75 million packets with the maximum packet arrival and the minimum processing rate of 0.20-0.25 million packets per second (mpps) and 0.30-0.35 mpps, respectively, and the maximum packet waiting time is 0.173-0.249 second.



## 9.1. Summary of the Thesis

---

In Chapter 5, two dynamic QoS-aware data traffic management schemes for the heterogeneous IoT applications in SD-DCN are presented considering the presence of heterogeneous flows. Firstly, we presented a game theory-based dynamic data traffic management scheme, named TROD, for maximizing network throughput in SD-DCN in the presence of IoT devices. We used an evolutionary game-theoretic approach to deciding the optimal data traffic volume which needs to be handled by the switches, while considering that the data generation rate for each IoT device is known a priori. Through simulations, we observed that TROD outperforms the existing schemes – Mobi-Flow and CURE, while distributing of data traffic among the available switches and reducing the volumetric overhead per switch by 23.4-29.7%. Thereafter, we introduced another QoS-aware stochastic data flow management scheme, named FlowMan, for SD-DCN in the presence of heterogeneous flows. We used a generalized Nash bargaining game to decide the Pareto optimal data rate to be allocated to each SDN switch, while considering the heterogeneous flows within the one-hop network. Further, using a distributed heuristic method, we decided the switch and flow-rule association for ensuring optimal data flow management in SD-DCN. Through simulations, we observed that FlowMan outperforms the existing benchmark schemes — CURE and FlowStat, while ensuring high throughput and low delay. In particular, FlowMan reduces network delay by 77.8–98.7% and increases network throughput by 24.6–47.8%, than using the existing schemes.

In Chapter 6, we presented a game-theory-based scheme, named D2B, for data broadcasting in SD-DCN in the presence of mobile IoT devices. We observed that the bandwidth distribution among the devices at the edge-tier of the fat-tree SD-DCN follows a leader-follower structure. Hence, we used a single-leader-multiple-follower Stackelberg game for designing the D2B scheme. We observed that D2B ensures the reduction in network delay in the presence of the mobile IoT devices at the edge-tier of the fat-tree SD-DCN. Moreover, from simulation, we observe that D2B outperforms the other existing schemes — LSBT and DCN\_INFOCOM. In particular, we observed that using

D2B, the network throughput increased by 55.32%, while ensuring at least 33% increase in the average bandwidth allocation per IoT device, and reduction in the overall delay.

In Chapter 7, we explored a single-leader-multiple-follower Stackelberg game for designing a dynamic data multicasting scheme, named D2M, in SD-DCN. In D2M, the controller takes strategic decision for rule placement among the switches while ensuring efficient load balancing. On the other hand, the switches are responsible for processing the data packets. Additionally, we considered that the switches are capable of bisecting the capacity into multiple resource blocks, where a subset of resource blocks needs to be allocated for each flow. From the simulation, we observed that D2M outperforms the other existing schemes — LSBT and BLO. In particular, we observed that using D2M, the network throughput increased by 6.13-95.32% than using existing schemes, while ensuring 21.32-99.29% reduction in delay.

In Chapter 8, we proposed BIND, a blockchain-based flow-table partitioning scheme, for distributed multi-tenant SDN. Using blockchain, we ensured that the controllers are synchronized and cooperative in nature. In BIND, we used utility game to propose the distributed algorithm for flow-rule election, where each controller distributively identifies the flow-rules' replacement eligibility factors, and elects a single flow-rule for replacement. Thereafter, we considered a utility game-based centralized algorithm for flow-rule replacement to be performed by the controller receiving the Packet-In message. We observed that BIND ensures fairness in flow-rule replacement for the controllers in a distributed multi-tenant SDN. Through simulation, we observed that the flow setup delay increases linearly using BIND. BIND also ensures high throughput and 100% network sustainability, thereby outperforming the benchmark schemes – HARD and PROXY. In particular, we observed that using BIND, the flow setup delay reduces by 48.1-49.7%, while ensuring 12.5-14.3% increase in network throughput than using the existing schemes.

## 9.2 Contributions

In this Thesis, traffic management schemes in the presence of heterogeneous flows were proposed for SD-DCN impeded by various challenges. The proposed schemes were designed to cope with the main issues – performance analysis, delay and throughput-optimal data traffic management, broadcasting and multicasting, and the presence of distributed multi-tenants, i.e., distributed multiple controllers. We list the major contributions of this Thesis as follows.

**Probabilistic Performance Analysis of OpenFlow Switch in SD-DCN:** We developed a Markovian model to replicate the behavior of an OpenFlow switch based on OpenFlow switch specification version 1.5.0 [5], when an incoming flow of packets passes through it. In our model, we considered multiple switches, instead of a single switch per controller. For each switch, we estimated the necessary performance metrics considering packet queuing, ingress and egress processing.

**Buffer Size Evaluation of OpenFlow Systems in SD-DCN:** We evaluated the optimum buffer size based on different performance metrics such as packet arrival and processing rates, and packet waiting time, in case of packet flow through an OpenFlow switch-based system. Initially, a queueing theory-based analytical scheme, named OPUS, based on the existing OpenFlow protocol [5] is developed. In OPUS, we designed the OpenFlow system as a I-M/M/1/K queue.

**Network-Specific QoS-Aware Dynamic Data Traffic Management in SD-DCN:** We introduced a game theory-based dynamic data traffic management scheme, named TROD, for minimizing network delay and maximizing network throughput in SD-DCN in the presence of IoT devices. We used an evolutionary game-theoretic approach to deciding the optimal data traffic volume which needs to be handled by the switches, while considering that the data generation rate for each IoT device is known a priori. In

TROD, the IoT devices act as the players and choose the set of optimal SDN switches, i.e., strategies in TROD, for forwarding data with the help of SDN controller.

### **Flow-Specific QoS-Aware Dynamic Data Traffic Management in SD-DCN:**

We designed FlowMan for delay-aware stochastic data flow management for SD-DCN in the presence of heterogeneous flows. We used a generalized Nash bargaining game to decide the Pareto optimal data rate to be allocated to each SDN-switch while considering the heterogeneous flows within the one-hop network. We considered that the switches act cooperatively to ensure high network throughput and low processing delay. In FlowMan, the strategy of each switch is to decide the optimal subset of flows to be handled by it.

### **Broadcast Data Traffic Management in Fat-Tree SD-DCN:**

We presented a dynamic bandwidth allocation scheme, named D2B, for data broadcasting in fat-tree SD-DCN, while considering the source node at the edge-tier to be mobile in nature. In D2B, we used a single-leader-multiple-follower Stackelberg game. Hence, prior to deciding the amount of bandwidth to be allocated to each node, each switch makes the list of connected nodes and the maximum capacity of the nodes. Based on this information, the control plane of each switch decides the amount of bandwidth to be allocated to each connected node for ensuring balanced load distribution in the fat-tree SD-DCN.

### **Multicast Data Traffic Management in Fat-Tree SD-DCN:**

We used a single-leader-multiple-follower Stackelberg game for designing a dynamic data multicasting scheme, named D2M, in SD-DCN. In D2M, we considered that the controller acts as the leader and installs the flow-rules in the SDN-switches. Additionally, the controller decides the source node of the flow for each destination. On the other hand, the SDN-switches, which act as the followers, decide their respective strategies, non-cooperatively. The followers help the controller to manage the network properly while deciding the

### 9.3. Limitations

---

amount of bandwidth to be allocated for each flow and optimize the usage of overall capacity.

**Flow-Table Partitioning in Distributed Multi-Tenant SD-DCN:** We introduced a utility game theory-based flow-table partitioning scheme, named BIND, for maximizing the network sustainability and minimizing the network overhead and delay in a distributed multi-tenant SDN. To ensure fairness among the multi-tenant controllers, we used blockchain-based flow-table partitioning. In BIND, we considered that the flow-tables are virtually owned by each controller.

### 9.3 Limitations

We made few assumptions while designing the proposed schemes.

- Packet processing at SDN switch can be designed as a Markovian process.
- Every event in packet processing is equally probable.
- Incoming packets follow a Poisson distribution.
- IoT devices are heterogeneous in nature and generate heterogeneous flows.
- Each edge between two nodes, i.e., IoT device or switch, has limited bandwidth.
- Source IoT devices are mobile in nature.
- Full coverage of the network is ensured.
- In multi-tenant SDN, there is no centralized SDN-controller, i.e., proxy controller, to coordinate.
- In multi-tenant SDN,, the SDN-controllers do not misbehave.

## 9.4 Future Scope of Work

The schemes proposed in this Thesis, can be extended while considering the following aspects.

- Future extension of AMOPE, which is designed for theoretical performance analysis of SDN system, includes proposing an efficient queuing scheme, so that queuing delay of packet flow gets reduced significantly. In addition, this work can be extended to understand how packet drop rate can be reduced while using the available TCAM memory in an OpenFlow switch.
- Future extension of OPUS, which is designed for buffer size analysis of SDN system, includes designing a scheme for improving the queueing model with multiple OpenFlow switches, and reducing waiting time or queueing delay in an OpenFlow system, while ensuring proper utilization of TCAM memory. This work also can be extended to visualize using SDN emulator such as Mininet, while considering real-time parameters – queuing delay for inter-switch communication and duration for flow-table update. In addition, this work can be extended to understand how the queueing model for group table functions in an OpenFlow switch-based system with proper utilization of TCAM memory.
- Future extension of the problems addressed in TROD and FlowMan, includes designing a data flow management scheme for broadcasting while minimizing the length of the routing path of the flows. The aforementioned problem can be mapped to the traveling salesman problem, which is an NP-hard problem. Hence, obtaining a Pareto optimal solution is challenging. Additionally, this work can be extended while incorporating the possibility of over-subscription in the presence of switches with limited flow-table capacity.
- Future extension of D2B, includes an understanding of network bandwidth distri-

#### 9.4. Future Scope of Work

---

bution in the presence of multiple source IoT devices at the edge-tier of fat-tree DCN. This work also can be extended to understand the optimal bandwidth distribution in the core and backhaul network. Additionally, this work can be extended to understand how network bandwidth is to be distributed while reducing the energy consumption of the network.

- Future extension of the problem addressed in D2M includes a study of the network bandwidth distribution in the presence of IoT devices with heterogeneous data/flows, i.e., elephant and mice flows, at the edge-tier of DCN. Additionally, this work can be extended to understand the interference among the flows if the same physical resources are shared among multiple flows. This work also can be extended to reduce the energy consumption of the network, while distributing the available bandwidth. Moreover, the last work, BIND, which is designed for multi-tenant SD-DCN system, can be extended by studying the energy consumption and flow-table aggregation among the controllers in a distributed multi-tenant SDN. Additionally, the effect of correlated flows can be explored. Furthermore, this work can be extended while considering that the controllers are non-cooperative in nature.
- The problems addressed in this thesis can also be revisited considering the provision for virtualization of multiple resources in network slicing for 5G-based IoT networks in the presence of SDN. Additionally, similar to the controller failure, the problems can be re-visited considering the effects of link failure.





# Publications

## Journals

- **A. Mondal** and S. Misra, “FlowMan: QoS-Aware Dynamic Data Flow Management in Software-Defined Networks”, *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1366-1373, July 2020. DOI: 10.1109/JSAC.2020.2999682
- S. Misra, **A. Mondal**, and S. Khajjayam, “Dynamic Big-Data Broadcast in Fat-Tree Data Center Networks with Mobile IoT Devices,” *IEEE Systems Journal*, vol. 13, no. 3, pp. 2898–2905, September 2019. DOI: 10.1109/JSYST.2019.2899754
- **A. Mondal**, S. Misra, and I. Maity, “AMOPE: Performance Analysis of OpenFlow Systems in Software-Defined Networks,” *IEEE Systems Journal*, vol. 14, no. 1, pp. 124-131, March 2020. DOI: 10.1109/JSYST.2019.2912843
- **A. Mondal**, S. Misra and I. Maity, “Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks,” *IEEE Systems Journal*, vol. 13, no. 2, pp. 1359–1366, June 2019. DOI: 10.1109/JSYST.2018.2820745

## Conferences

- **Ayan Mondal** and Sudip Misra, “BIND: Blockchain-Based Flow-Table Partitioning in Distributed Multi-Tenant Software-Defined Networks,” in *Proceedings of IEEE International Conference on Computer Communications Workshops (INFOCOM Workshops)*, Toronto, Canada, July 2020, pp. 1-6. DOI: 10.1109/INFOCOMWKSHP50562.2020.9162868
- S. Misra, **A. Mondal**, and P. Kumar, “D2M: Mobility-Aware Dynamic Data Multicasting in Software-Defined Data Center Networks,” in *Proceedings of IEEE International Conference on Communications Workshops (ICC Workshops)*, Shanghai, China, 2019, pp. 1–6. DOI: 10.1109/ICCW.2019.8756718  
(Hot Topic Paper Award)
- **A. Mondal**, S. Misra and A. Chakraborty, “TROD: Throughput-Optimal Dynamic Data Traffic Management in Software-Defined Networks,” in *Proceedings of IEEE Global Communications Conference Workshops (GLOBECOM Workshops)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1–6. DOI: 10.1109/GLOCOMW.2018.8644398



# References

- [1] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, “Big Data and Its Technical Challenges,” *Communications of the ACM Magazine*, vol. 57, no. 7, pp. 86–94, July 2014.
- [2] J. Choo and H. Park, “Customizing Computational Methods for Visual Analytics with Big Data,” *IEEE Computer Graphics and Applications*, vol. 33, no. 4, pp. 22–28, July 2013.
- [3] G. Andrienko, N. Andrienko, and S. Wrobel, “Visual Analytics Tools for Analysis of Movement Data,” *ACM SIGKDD*, vol. 9, no. 2, pp. 38–46, Dec. 2007.
- [4] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, 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, January 2015.
- [5] OpenFlow. (2014, December) OpenFlow Switch Specification Version 1.5.0. Open Networking Foundation. [Online]. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>
- [6] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, “Simultaneously Reducing Latency and Power Consumption in OpenFlow Switches,” *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 1007–1020, June 2014.
- [7] N. P. Katta, J. Rexford, and D. Walker, “Incremental Consistent Updates,” in *Proc. of ACM SIGCOMM Wrkshp.* New York, NY, USA: ACM, 2013, pp. 49–54.
- [8] C. R. Meiners, A. X. Liu, and E. Torng, “Bit Weaving: A Non-Prefix Approach to Compressing Packet Classifiers in TCAMs,” *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 488–500, April 2012.
- [9] 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 *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, December 2013, pp. 1397–1402.

- 
- [10] 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 23<sup>rd</sup> International Teletraffic Congress*. International Teletraffic Congress, 2011, pp. 1–7.
- [11] M. Chen, S. Mao, and Y. Liu, "Big Data: A Survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.
- [12] G. M. Muntean, P. Perry, and L. Murphy, "Subjective Assessment of the Quality-Oriented Adaptive Scheme," *IEEE Transactions on Broadcasting*, vol. 51, no. 3, pp. 276–286, September 2005.
- [13] C. J. Wu, C. F. Ku, J. M. Ho, and M. S. Chen, "A Novel Pipeline Approach for Efficient Big Data Broadcasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 17–28, January 2016.
- [14] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou, "Networking for Big Data: A Survey," *IEEE Communications Surveys Tutorials*, 2016, DOI: 10.1109/COMST.2016.2610963.
- [15] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "DevoFlow: Cost-effective Flow Management for High Performance Enterprise Networks," in *Proceedings of the 9<sup>th</sup> ACM SIGCOMM Workshop on Hot Topics in Networks*, New York, NY, USA, 2010, pp. 1–6.
- [16] C. Metter, M. Seufert, F. Wamser, T. Zinner, and P. Tran-Gia, "Analytical Model for SDN Signaling Traffic and Flow Table Occupancy and Its Application for Various Types of Traffic," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 603–615, Sep 2017.
- [17] 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," in *Proceedings of ACM SIGCOMM Conference on Data Communication*, New York, NY, USA, 2009, pp. 51–62.
- [18] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proceedings of ACM SIGCOMM Conference on Data Communication*, New York, NY, USA, 2008, pp. 63–74.
- [19] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling Flow Management for High-performance Networks," in *Proceedings of ACM SIGCOMM Conference*, New York, NY, USA, 2011, pp. 254–265.
- [20] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, "Decentralized Task-aware Scheduling for Data Center Networks," in *Proc. of ACM SIGCOMM*, Aug. 2014, pp. 431–442.

## References

---

- [21] Z. Guo and Y. Yang, "Multicast Fat-Tree Data Center Networks with Bounded Link Oversubscription," in *Proceedings of IEEE INFOCOM*, April 2013, pp. 350–354.
- [22] B. Ciubotaru, C. H. Muntean, and G. M. Muntean, "Mobile Multi-Source High Quality Multimedia Delivery Scheme," *IEEE Transactions on Broadcasting*, vol. 63, no. 2, pp. 391–403, June 2017.
- [23] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 655–685, Firstquarter 2016.
- [24] M. Caria, A. Jukan, and M. Hoffmann, "Sdn partitioning: A centralized control plane for distributed routing protocols," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 381–393, Sep. 2016.
- [25] Y. Lin, T. Liu, J. Chen, and Y. Lai, "Soft partitioning flow tables for virtual networking in multi-tenant software defined networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 402–415, March 2018.
- [26] D. L. Eager and K. C. Sevcik, "Performance Bound Hierarchies for Queueing Networks," *ACM Transactions on Computer Systems*, vol. 1, no. 2, pp. 99–115, May 1983.
- [27] P. Garrido, D. E. Lucani, and R. AgÃijero, "Markov Chain Model for the Decoding Probability of Sparse Network Coding," *IEEE Transactions on Communications*, vol. 65, no. 4, pp. 1675–1685, April 2017.
- [28] P. D. Bergstrom, M. A. Ingram, A. J. Vernon, J. L. A. Hughes, and P. Tetali, "A markov chain model for an optical shared-memory packet switch," *IEEE Transactions on Communications*, vol. 47, no. 10, pp. 1593–1603, October 1999.
- [29] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow Switching: Data Plane Performance," in *Proceedings of IEEE International Conference on Communications*, May 2010, pp. 1–5.
- [30] M. Rich and M. Schwartz, "Buffer Sharing in Computer-Communication Network Nodes," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 958–970, September 1977.
- [31] H. Kekre and C. Saxena, "Finite buffer behavior with poisson arrivals and random server interruptions," *IEEE Transactions on Communications*, vol. 26, no. 4, pp. 470–474, Apr 1978.
- [32] G. Luan, "Buffer Stopping Time Analysis in Data Center Networks," *IEEE Comm. Let.*, vol. 18, no. 10, pp. 1739–1742, October 2014.
- [33] M. Cello, G. Gnecco, M. Marchese, and M. Sanguineti, "A Model of Buffer Occupancy for ICNs," *IEEE Comm. Let.*, vol. 16, no. 6, pp. 862–865, June 2012.

- 
- [34] B. R. Manoj, R. K. Mallik, and M. R. Bhatnagar, "Buffer-Aided Multi-Hop DF Cooperative Networks: A State-Clustering Based Approach," *IEEE Trans. on Comm.*, vol. 64, no. 12, pp. 4997–5010, December 2016.
- [35] A. Asheralieva and Y. Miyanaga, "Dynamic Buffer Status-Based Control for LTE-A Network With Underlay D2D Communication," *IEEE Transactions on Communications*, vol. 64, no. 3, pp. 1342–1355, March 2016.
- [36] K. Jagannathan, M. Markakis, E. Modiano, and J. N. Tsitsiklis, "Queue-Length Asymptotics for Generalized Max-Weight Scheduling in the Presence of Heavy-Tailed Traffic," *IEEE/ACM Trans. on Net.*, vol. 20, no. 4, pp. 1096–1111, August 2012.
- [37] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for Network Update," in *Proceedings of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, NY, USA, 2012, pp. 323–334.
- [38] S. Bera, S. Misra, and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, 2017.
- [39] N. Saha, S. Misra, and S. Bera, "QoS-Aware Adaptive Flow-Rule Aggregation in Software-Defined IoT," in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 206–212.
- [40] I. Maity, A. Mondal, S. Misra, and C. Mandal, "Tensor-Based Rule-Space Management System in SDN," *IEEE Systems Journal*, vol. 13, no. 4, pp. 3921–3928, 2019.
- [41] H. Huang, S. Guo, P. Li, B. Ye, and I. Stojmenovic, "Joint Optimization of Rule Placement and Traffic Engineering for QoS Provisioning in Software Defined Network," *IEEE Trans. on Comp.*, vol. 64, no. 12, pp. 3488–3499, Dec 2015.
- [42] Y. Sadeh, O. Rottenstreich, A. Barkan, Y. Kanizo, and H. Kaplan, "Optimal Representations of a Traffic Distribution in Switch Memories," in *Proc. of IEEE INFOCOM*, Apr 2019, pp. 2035–2043.
- [43] O. Rottenstreich, Y. Kanizo, H. Kaplan, and J. Rexford, "Accurate Traffic Splitting on SDN Switches," *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 10, pp. 2190–2201, Oct 2018.
- [44] M.-H. Wang, P.-W. Chi, J.-W. Guo, and C.-L. Lei, "SDN storage: A Stream-Based Storage System over Software-Defined Networks," in *Proc. of IEEE INFOCOM Wrksp*s, Apr 2016, pp. 598–599.
- [45] F. Li, J. Cao, X. Wang, Y. Sun, T. Pan, and X. Liu, "Adopting SDN Switch Buffer: Benefits Analysis and Mechanism Design," in *Proc. of IEEE ICDCS*, Jun 2017, pp. 2171–2176.

## References

---

- [46] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, “Scalable Architecture for SDN Traffic Classification,” *IEEE Syst. J.*, pp. 1–12, 2017, DOI: 10.1109/JSYST.2017.2690259.
- [47] N. Saha, S. Bera, and S. Misra, “Sway: Traffic-Aware QoS Routing in Software-Defined IoT,” *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2018.
- [48] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, “Soft-WSN: Software-Defined WSN Management System for IoT Applications,” *IEEE Syst. J.*, pp. 1–8, 2016, DOI:10.1109/JSYST.2016.2615761.
- [49] S. Misra, S. Bera, A. M. P., S. K. Pal, and M. S. Obaidat, “Situation-Aware Protocol Switching in Software-Defined Wireless Sensor Network Systems,” *IEEE Systems Journal*, vol. 12, no. 3, pp. 2353–2360, 2018.
- [50] S. Bera, S. Misra, and M. S. Obaidat, “Mobility-Aware Flow-Table Implementation in Software-Defined IoT,” in *Proc. of IEEE GLOBECOM*, December 2016, pp. 1–6.
- [51] S. Bera, S. Misra, and M. S. Obaidat, “Mobi-Flow: Mobility-Aware Adaptive Flow-Rule Placement in Software-Defined Access Network,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1831–1842, 2019.
- [52] S. Agarwal, M. Kodialam, and T. V. Lakshman, “Traffic Engineering in Software Defined Networks,” in *Proc. of IEEE INFOCOM*, Apr 2013, pp. 1–9.
- [53] S.-H. Tseng, A. Tang, G. L. Choudhury, and S. Tse, “Routing Stability in Hybrid Software-Defined Networks,” *IEEE/ACM Trans. on Net.*, vol. 27, no. 2, pp. 790–804, Apr 2019.
- [54] S. Misra and S. Bera, “Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2071–2078, 2020.
- [55] S. Misra and N. Saha, “Detour: Dynamic Task Offloading in Software-Defined Fog for IoT Applications,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.
- [56] M. Moradi, Y. Zhang, Z. Morley Mao, and R. Manghirmalani, “Dragon: Scalable, Flexible, and Efficient Traffic Engineering in Software Defined ISP Networks,” *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 12, pp. 2744–2756, Dec 2018.
- [57] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi, “Multi-Path Alpha-Fair Resource Allocation at Scale in Distributed Software-Defined Networks,” *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 12, pp. 2655–2666, Dec 2018.
- [58] D. Sanvito, I. Filippini, A. Capone, S. Paris, and J. Leguay, “Adaptive Robust Traffic Engineering in Software Defined Networks,” in *Proc. of IFIP Networking and Workshops*, May 2018, pp. 1–9.

- 
- [59] A. Mondal, S. Misra, and A. Chakraborty, "TROD: Throughput-Optimal Dynamic Data Traffic Management in Software-Defined Networks," in *Proc. of IEEE Globecom Workshops*, Dec 2018, pp. 1–6.
- [60] H. Tahaei, R. B. Salleh, M. F. A. Razak, K. Ko, and N. B. Anuar, "Cost Effective Network Flow Measurement for Software Defined Networks: A Distributed Controller Scenario," *IEEE Access*, vol. 6, pp. 5182–5198, 2018.
- [61] B. G urkemli, S. Tatlicioglu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic Control Plane for SDN at Scale," *IEEE J. on Sel. Areas in Comm.*, vol. 36, no. 12, pp. 2688–2701, Dec 2018.
- [62] S. Bera, S. Misra, and N. Saha, "DynamITE: Dynamic Traffic Engineering in Software-Defined Cyber Physical Systems," in *Proc. of IEEE ICC Workshops*, May 2018, pp. 1–6.
- [63] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An Intelligent Route Computation Approach Based on Real-Time Deep Learning Strategy for Software Defined Communication Systems," *IEEE Trans. on Emerg. Topics in Comp.*, pp. 1–12, 2019.
- [64] O. Rottenstreich, I. Keslassy, Y. Revah, and A. Kadosh, "Minimizing Delay in Network Function Virtualization with Shared Pipelines," *IEEE Trans. on Par. and Dist. Syst.*, vol. 28, no. 1, pp. 156–169, Jan 2017.
- [65] O. Rottenstreich and J. Tapolcai, "Lossy Compression of Packet Classifiers," in *Proc. of ACM/IEEE ANCS*, May 2015, pp. 39–50.
- [66] A. Singh, S. Batra, G. S. S. Aujla, N. Kumar, and L. T. Yang, "Bloom-Store: Dynamic Bloom Filter-based Secure Rule-Space Management Scheme in SDN," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2020, DOI: 10.1109/TII.2020.2966708.
- [67] G. S. Aujla, R. Chaudhary, N. Kumar, R. Kumar, and J. J. P. C. Rodrigues, "An Ensembled Scheme for QoS-Aware Traffic Flow Management in Software Defined Networks," in *Proc. of IEEE Int. Conf. on Comm.*, May 2018, pp. 1–7.
- [68] S. Bera, S. Misra, and N. Saha, "Traffic-aware Dynamic Controller Assignment in SDN," *IEEE Transactions on Communications*, pp. 1–8, 2020, DOI: 10.1109/TCOMM.2020.2983168.
- [69] Z. Guo and Y. Yang, "On Nonblocking Multicast Fat-Tree Data Center Networks with Server Redundancy," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 1058–1073, April 2015.
- [70] W. Liu, K. Nakauchi, and Y. Shoji, "A Neighbor-Based Probabilistic Broadcast Protocol for Data Dissemination in Mobile IoT Networks," *IEEE Access*, vol. 6, pp. 12 260–12 268, 2018.



## References

---

- [71] C. P. Lau, A. Alabbasi, and B. Shihada, “An Efficient Live TV Scheduling System for 4G LTE Broadcast,” *IEEE Syst. J.*, vol. 11, no. 4, pp. 2737–2748, Dec. 2017.
- [72] T. Zarb and C. J. Debono, “Broadcasting Free-Viewpoint Television Over Long-Term Evolution Networks,” *IEEE Syst. J.*, vol. 10, no. 2, pp. 773–784, Jun. 2016.
- [73] H. Lakhlef, A. Bouabdallah, M. Raynal, and J. Bourgeois, “Agent-Based Broadcast Protocols for Wireless Heterogeneous Node Networks,” *Comp. Comm.*, vol. 115, pp. 51 – 63, 2018.
- [74] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, “A Survey of Information-Centric Networking,” *IEEE Comm. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
- [75] R. Trestian, O. Ormond, and G. M. Muntean, “Enhanced Power-Friendly Access Network Selection Strategy for Multimedia Delivery Over Heterogeneous Wireless Networks,” *IEEE Transactions on Broadcasting*, vol. 60, no. 1, pp. 85–101, March 2014.
- [76] D. Paul, W. D. Zhong, and S. K. Bose, “Demand Response in Data Centers Through Energy-Efficient Scheduling and Simple Incentivization,” *IEEE Syst. J.*, vol. 11, no. 2, pp. 613–624, Jun. 2017.
- [77] A. Iyer, P. Kumar, and V. Mann, “Avalanche: Data Center Multicast using Software Defined Networking,” in *Proceedings of the Sixth International Conference on Communication Systems and Networks (COMSNETS)*, 2014, pp. 1–8.
- [78] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, “Improving Datacenter Performance and Robustness with Multipath TCP,” in *Proceedings of ACM SIGCOMM Conference*, New York, NY, USA, 2011, pp. 266–277.
- [79] R. Zhu, D. Niu, B. Li, and Z. Li, “Optimal multicast in virtualized datacenter networks with software switches,” in *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.
- [80] E. Chiu and V. K. N. Lau, “Precoding Design for Multi-Antenna Multicast Broadcast Services With Limited Feedback,” *IEEE Syst. J.*, vol. 4, no. 4, pp. 550–560, Dec. 2010.
- [81] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks,” in *Proc. of USENIX Conf. on Net. Sys. Des. and Impl.*, Mar. 2010, pp. 1–15.
- [82] A. R. Curtis, S. Keshav, and A. Lopez-Ortiz, “LEGUP: Using Heterogeneity to Reduce the Cost of Data Center Network Upgrades,” in *Proc. of ACM SIGCOMM*, Aug. 2010, pp. 1–12.

- 
- [83] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535–547, March 2000.
- [84] A. Markov, "Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain," in *Dynamic Probabilistic Systems (Volume I: Markov Models)*. John Wiley & Sons, Inc., 1971, pp. 552–577.
- [85] M. A. Marsan, G. Conte, and G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Trans. on Comp. Sys.*, vol. 2, no. 2, pp. 93–122, May 1984.
- [86] J. Suzuki, "A Markov Chain Analysis on Simple Genetic Algorithms," *IEEE Trans. on Syst., Man, and Cyb.*, vol. 25, no. 4, pp. 655–659, April 1995.
- [87] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An Open Framework for OpenFlow Switch Evaluation," in *Proceedings of International Conference on Passive and Active Network Measurement*. Springer, 2012, pp. 85–95.
- [88] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. Wiley-Interscience, 1975.
- [89] N. Sapountzis, T. Spyropoulos, N. Nikaiein, and U. Salim, "Optimal Downlink and Uplink User Association in Backhaul-Limited HetNets," in *Proc. of IEEE INFOCOM*, April 2016, pp. 1–9.
- [90] S. Misra and A. Chakraborty, "QoS-Aware Dispersed Dynamic Mapping of Virtual Sensors in Sensor-Cloud," *IEEE Transactions on Services Computing*, pp. 1–12, May 2019, DOI: 10.1109/TSC.2019.2917447.
- [91] H. Park and M. van der Schaar, "Bargaining Strategies for Networked Multimedia Resource Management," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3496–3511, July 2007.
- [92] J. D. Ullman, "NP-Complete Scheduling Problems," *J. of Com. & Syst. Sc.*, vol. 10, no. 3, pp. 384–393, 1975.
- [93] I. Maity, A. Mondal, S. Misra, and C. Mandal, "CURE: Consistent Update with Redundancy Reduction in SDN," *IEEE Trans. on Comm.*, vol. PP, no. 99, pp. 1–8, 2018.
- [94] Federal Communications Commission (FCC). (2019, Aug.) Broadband Speed Guide. [Online]. Available: <https://www.fcc.gov/reports-research/guides/broadband-speed-guide>
- [95] R. Trestian, K. Katrinis, and G. Muntean, "OFLoad: An OpenFlow-Based Dynamic Load Balancing Strategy for Datacenter Networks," *IEEE Trans. on Net. Serv. Man.*, vol. 14, no. 4, pp. 792–803, Dec 2017.

## References

---

- [96] A. Mondal, S. Misra, and I. Maity, “Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks,” *IEEE Syst. J.*, pp. 1–8, 2018.
- [97] A. Drexler, “A Simulated Annealing Approach to the Multiconstraint Zero-One Knapsack Problem,” *Computing*, vol. 40, no. 1, pp. 1–8, Mar 1988.
- [98] R. V. Driessche and D. Roose, “An Improved Spectral Bisection Algorithm and Its Application to Dynamic Load Balancing,” *Parallel Computing*, vol. 21, no. 1, pp. 29 – 48, 1995.
- [99] V. Poirriez, N. Yanev, and R. Andonov, “A Hybrid Algorithm for the Unbounded Knapsack Problem,” *Discrete Optimization*, vol. 6, no. 1, pp. 110 – 124, 2009.
- [100] S. Bera, S. Misra, and A. Jamalipour, “FlowStat: Adaptive Flow-Rule Placement for Per-Flow Statistics in SDN,” *IEEE J. on Sel. Areas in Comm.*, vol. 37, no. 3, pp. 530–539, March 2019.
- [101] R. W. D. Nickalls, “A New Approach to Solving the Cubic: Cardan’s Solution Revealed,” *The Math. Gaz.*, vol. 77, no. 480, pp. 354–359, 1993.
- [102] NB-IoT – Enabling New Business Opportunities. HUAWEI. [Accessed on: May 10, 2018]. [Online]. Available: [http://www.huawei.com/minisite/iot/img/nb\\_iiot\\_whitepaper\\_en.pdf](http://www.huawei.com/minisite/iot/img/nb_iiot_whitepaper_en.pdf)
- [103] S. Saroiu, P. K. Gummadi, and S. D. Gribble, “Measurement Study of Peer-to-Peer File Sharing Systems,” in *Proceedings of International Society for Optics and Photonics - Multimedia Computing and Networking*, vol. 4673. International Society for Optics and Photonics, 2002, pp. 156–170.
- [104] B. Liang and Z. J. Haas, “Predictive distance-based mobility management for PCS networks,” in *Proc. of IEEE INFOCOM*, vol. 3, Mar. 1999, pp. 1377–1384.
- [105] D. B. Johnson and D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, 1996, pp. 153–181.
- [106] R. W. Rosenthal, “Games of perfect information, predatory pricing and the chain-store paradox,” *Journal of Economic Theory*, vol. 25, no. 1, pp. 92–100, Aug. 1981.
- [107] K. Christidis and M. Devetsikiotis, “Blockchains and Smart Contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
- [108] A. Chakraborty, A. Mondal, and S. Misra, “Cache-Enabled Sensor-Cloud: The Economic Facet,” in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, Apr 2018, pp. 1–6.
- [109] A. Chakraborty, S. Misra, A. Mondal, and M. S. Obaidat, “SensOrch: QoS-Aware Resource Orchestration for Provisioning Sensors-as-a-Service,” in *Proceedings of IEEE Conference on Communications (ICC)*, 2020, pp. 1–6.

- [110] D. Joseph and D. Grunwald, "Prefetching using Markov Predictors," *IEEE Transactions on Computers*, vol. 48, no. 2, pp. 121–133, Feb 1999.

## BIO-DATA

### 1. Bio-data:

- *Name:* Mr. Ayan Mondal
- *Roll No.:* 14IT92P07
- *Father's Name:* Mr. Bhajahari Mondal
- *Mother's Name:* Mrs. Sandhya Mondal
- *Date of Birth:* 12<sup>th</sup> January, 1991
- *Permanent Address:* Ashrampara (Teachers' Colony), Basirhat  
P.O. - Basirhat, Dist. - North 24 Parganas  
State - West Bengal, India, PIN - 743411
- ✉ ayanmondal@iitkgp.ac.in; mondalayan12@gmail.com

2. **Present Status:** PhD Research Scholar, TCS Fellow (India)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur, India

### 3. Academic Qualification:

- **Master of Science (by Research)** 2015  
School of Information Technology  
Indian Institute of Technology Kharagpur  
**Thesis:** *Distributed Energy Management in Smart Grid*  
**Supervisor:** Professor Sudip Misra  
**Grade (out of 10):** 9.73
- **Bachelor of Technology** 2012  
Electronics and Communication Engineering  
West Bengal University of Technology  
**Grade (out of 10):** 8.99
- **Higher Secondary (Science)** 2008  
West Bengal Council of Higher Secondary Education  
**Percentage:** 84.57
- **Secondary** 2006  
West Bengal Board of Secondary Education  
**Percentage:** 92.00

#### 4. Research Experience:

- **TCS Research Fellow**, IIT Kharagpur  
Sponsored by DRDO Mar. 2016 – Jan. 2020
- **Visiting Researcher**, Inria, CNRS, IRISA  
Sponsored by Inria, Rennes, France Jun. 2019 – Sep. 2019
- **Senior Research Fellow**, IIT Kharagpur  
Sponsored by DRDO, Govt. of India Feb. 2016
- **Senior Project Officer**, IIT Kharagpur  
Sponsored by DietY, Govt. of India Aug. 2014 – Jan. 2016
- **Junior Project Officer**, IIT Kharagpur  
Sponsored by DietY, Govt. of India Aug. 2013 – Jul. 2014
- **Junior Project Assistant**, IIT Kharagpur  
Sponsored by DietY, Govt. of India Aug. 2012 – Jul. 2013

#### 5. Teaching Assistance:

- Software Engineering Lab (CS29006)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jan. 2020 - Jul. 2020
- Software Engineering Theory (CS20006)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jan. 2020 - Jul. 2020
- Software Engineering Lab (CS29006)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jan. 2019 - Jun. 2019
- Software Engineering Theory (CS20006)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jan. 2019 - Jul. 2019
- Programming and Data Structures Lab (CS19001)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jul. 2018 - Dec. 2018
- Software Engineering Lab (CS29006)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jan. 2018 - Jun. 2018
- Wireless Ad-Hoc and Sensor Networks (IT60119)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jul. 2017 - Dec. 2017
- Programming and Data Structures Lab (CS19001)  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur Jan. 2017 - Jun. 2017

- Wireless Ad-Hoc and Sensor Networks (IT60119) Jul. 2016 - Dec. 2016  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur
- Internet and Web-based Technologies (IT60102) Jan. 2016 – May 2016  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur
- Underwater Sensor Networks: Theory and Simulations Apr. 2016  
Short-term Course  
NPOL, DRDO  
Sponsored by: DRDO

## 6. Publications:

### Journals

- [J19] S. Misra, **A. Mondal**, P. V. Sudheer Kumar, and Sankar K. Pal, "SEED: QoS-Aware Sustainable Energy Distribution in Smart Grid," *IEEE Transactions on Sustainable Computing*, pp.1-11, October 2020. [Manuscript ID: TSUSC-2020-04-0031.R1] (Accepted)
- [J18] A. Chakraborty, S. Misra, and **A. Mondal**, "QoS-Aware Dynamic Cost Management Scheme for Sensors-as-a-Service," *IEEE Transactions on Services Computing*, Early Access, pp. 1-12, July 2020. DOI: 10.1109/TSC.2020.3011495
- [J17] S. Misra, **A. Mondal**, P. Bhavathankar, and M.-S. Alouini, "M-JAW: Mobility-Based Jamming Avoidance in Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5381-5390, May 2020. DOI: 10.1109/TVT.2020.2982966
- [J16] **A. Mondal** and S. Misra, "FlowMan: QoS-Aware Dynamic Data Flow Management in Software-Defined Networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1366-1373, July 2020. DOI: 10.1109/JSAC.2020.2999682
- [J15] A. Roy, **A. Mondal**, S. Misra, and M. S. Obaidat, "ORCID: Opportunistic Re-Connectivity for Network Management in the Presence of Dumb Nodes in Wireless Sensor Networks," *IEEE Systems Journal*, vol. 14, no. 1, pp. 9-16, March 2020. DOI: 10.1109/JSYST.2019.2956324
- [J14] **A. Mondal**, S. Misra, and I. Maity, "AMOPe: Performance Analysis of OpenFlow Systems in Software-Defined Networks," *IEEE Systems Journal*, vol. 14, no. 1, pp. 124-131, March 2020. DOI: 10.1109/JSYST.2019.2912843
- [J13] S. Misra, **A. Mondal**, and S. Khajjayam, "Dynamic Big-Data Broadcast in Fat-Tree Data Center Networks with Mobile IoT Devices," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2898-2905, September 2019. DOI: 10.1109/JSYST.2019.2899754
- [J12] I. Maity, **A. Mondal**, S. Misra, and C. Mandal, "Tensor-Based Rule-Space Management System in SDN," *IEEE Systems Journal*, vol. 13, no. 4, pp. 3921-3928, December 2019. DOI: 10.1109/JSYST.2018.2879321

- [J11] A. Chakraborty, **A. Mondal**, A. Roy, and S. Misra, "Dynamic Trust Enforcing Pricing Scheme for Sensors-as-a-Service in Sensor-Cloud Infrastructure," *IEEE Transactions on Services Computing*, pp. 1-12, September 2018. DOI: 10.1109/TSC.2018.2873763
- [J10] I. Maity, **A. Mondal**, S. Misra, and C. Mandal, "CURE: Consistent Update with Redundancy Reduction in SDN," *IEEE Transactions on Communications*, vol. 66, no. 9, pp. 3974-3981, September 2018.
- [J9] **A. Mondal**, S. Misra, and I. Maity, "Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1359-1366, June 2019. DOI: 10.1109/JSYST.2018.2820745.
- [J8] **A. Mondal**, S. Misra, L. S. Patel, S. K. Pal and M. S. Obaidat, "DEMANDS: Distributed Energy Management Using Non-cooperative Scheduling in Smart Grid," *IEEE Systems Journal*, Vol. 12, no. 3, pp. 2645-2653, September 2018. DOI: 10.1109/JSYST.2017.2723961.
- [J7] P. Bhavathankar, **A. Mondal**, and S. Misra, "Topology Control in the Presence of Jammers for Wireless Sensor Networks," *International Journal of Communication Systems*, Vol. 30, no. 13, pp. 1-11, January 2017. DOI: 10.1002/dac.3289
- [J6] A. Roy, S. Misra, P. Kar, and **A. Mondal**, "Topology Control for Self-Adaptation in Wireless Sensor Networks with Temporary Connection Impairment," *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 11, no. 4, pp. 21:1-21:34, January 2017. DOI: 10.1145/2979680
- [J5] **A. Mondal**, S. Misra, and Mohammad S. Obaidat, "Distributed Home Energy Management System with Storage in Smart Grid Using Game Theory," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1857-1866, September 2017. DOI: 10.1109/JSYST.2015.2421941
- [J4] **A. Mondal** and S. Misra, "Game-Theoretic Energy Trading Network Topology Control for Electric Vehicles in Mobile Smart Grid," *IET Networks*, vol. 4, no. 4, pp. 220-228, July 2015. DOI: 10.1049/iet-net.2014.0089
- [J3] S. Misra, T. Ojha, and **A. Mondal**, "Game-theoretic Topology Control for Opportunistic Localization in Sparse Underwater Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 5, pp. 990-1003, July 2014.
- [J2] S. Misra, G. Mali, and **A. Mondal**, "Distributed Topology Management for Wireless Multimedia Sensor Networks: Exploiting Connectivity and Cooperation," *International Journal of Communication Systems*, vol. 27, no. 3, pp. 1367-1387, March 2014. DOI: 10.1002/dac.2770
- [J1] S. Misra, S. Bera, **A. Mondal**, R. Tirkey, H.-C. Chao, S. Chattopadhyay, "Optimal Gateway Selection in Sensor-Cloud Framework for Health Monitoring," *IET Wireless Sensor Systems*, vol. 3, no. 4, pp. 61-68, December 2013. DOI: 10.1049/iet-wss.2013.0073



## Conferences

- [C14] **A. Mondal** and S. Misra, "BIND: Blockchain-Based Flow-Table Partitioning in Distributed Multi-Tenant Software-Defined Networks," in *Proceedings of IEEE International Conference on Computer Communications Workshops (INFOCOM Workshops): Blockchain for Secure Software defined Networking in Smart Communities (BlockSecSDN)*, Toronto, Canada, July 2020, pp. 1-6. (Accepted)
- [C13] A. Chakraborty, S. Misra, **A. Mondal**, and Mohammad S. Obaidat, "SensOrch: QoS-Aware Resource Orchestration for Provisioning Sensors-as-a-Service," in *Proceedings of IEEE International Conference on Communications (ICC)*, Dublin, Ireland, June 2020, pp. 1-6. (Accepted)
- [C12] S. Misra, **A. Mondal**, and P. Kumar, "D2M: Mobility-Aware Dynamic Data Multicasting in Software-Defined Data Center Networks," in *Proceedings of IEEE International Conference on Communications Workshops (ICC Workshops): Secure and Dependable Software Defined Networking for Sustainable Smart Communities (SecSDN)*, Shanghai, China, March 2019, pp. 1-6. DOI: 10.1109/ICCW.2019.8756718 (**Hot Topic Paper Award**)
- [C11] S. Misra, A. Mondal, and **A. Mondal**, "DATUM: Dynamic Topology Control for Underwater Wireless Multimedia Sensor Networks," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakech, Morocco, December 2019, pp. 1-6. DOI: 10.1109/WCNC.2019.8885632
- [C10] **A. Mondal**, S. Misra, and A. Chakraborty, "TROD: Throughput-Optimal Dynamic Data Traffic Management in Software-Defined Networks," in *Proceedings of IEEE Global Communications Conference Workshops (GLOBECOM Workshops): Software defined Networking for 5G Architecture in Smart Communities*, Abu Dhabi, UAE, December 2018, pp. 1-6. DOI: 10.1109/GLOCOMW.2018.8644398
- [C9] **A. Mondal** and S. Misra, "Dynamic Micro-Grid Selection by Plug-In Electric Vehicles in Smart Grid: An Evolutionary Game," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Catalonia, Spain, April 2018, pp. 1-2. [Student Poster]
- [C8] A. Chakraborty, **A. Mondal**, and S. Misra, "Cache-Enabled Sensor-Cloud: The Economic Facet," in *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, Catalonia, Spain, April 2018, pp. 1-6. DOI: 10.1109/WCNC.2018.8377069
- [C7] **A. Mondal** and S. Misra, "DCoE: Game-Theoretic Dynamic Coalition Extension with Micro-Grid Failure in Smart Grid," in *Proceedings of IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Bhubaneswar, India, December 2017, pp. 1-6. DOI: 10.1109/ANTS.2017.8384118

- [C6] **A. Mondal** and S. Misra, "Game-theoretic Green Electric Vehicle Energy Networks Management in Smart Grid," in *Proceedings of IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Kolkata, India, December 2015, pp. 1-6. DOI: 10.1109/ANTS.2015.7413616
- [C5] **A. Mondal** and S. Misra, "Dynamic Data Aggregator Unit Selection in Smart Grid: An Evolutionary Game Theoretic Approach," in *Proceedings of the 11th IEEE India Conference on Emerging Trends and Innovation in Technology (INDICON)*, Pune, India, December 2014, pp. 1-6. DOI: 10.1109/INDICON.2014.7030614
- [C4] **A. Mondal** and S. Misra, "Game-Theoretic Distributed Virtual Energy Cloud Topology Control for Mobile Smart Grid," in *Proceedings of the 6<sup>th</sup> IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, December 2014, pp. 54-61. DOI: 10.1109/CloudCom.2014.83
- [C3] A. Roy, **A. Mondal**, and S. Misra, "Connectivity Re-establishment in the Presence of Dumb Nodes in Sensor-Cloud Infrastructure: A Game Theoretic Approach," in *Proceedings of Emerging Issues in Cloud (EIC) workshop in conjunction with the 6<sup>th</sup> IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, December 2014, pp. 847-852. DOI: 10.1109/CloudCom.2014.121
- [C2] S. Misra, **A. Mondal**, S. Banik, M. Khatua, S. Bera, and Mohammad S. Obaidat, "Residential Energy Management in Smart Grid: A Markov Decision Process-Based Approach," in *Proceedings of IEEE International Conference on Internet of Things (iThings)*, Beijing, China, August 2013, pp. 1152-1157. DOI: 10.1109/GreenCom-iThings-CPSCOM.2013.200
- [C1] **A. Mondal** and S. Misra, "Dynamic Coalition Formation in a Smart Grid: A Game Theoretic Approach," in *Proceedings of IEEE International Conference on Communications Workshops (ICC Workshops): Smart Communication Protocols and Algorithms (SCPA)*, Budapest, Hungary, June 2013, pp. 1067-1071. DOI: 10.1109/ICCW.2013.6649395

## 7. Patent:

- [P1] **A. Mondal**, S. K. Roy, A. Roy, and S. Misra, "A Cloud Based Automatized System for On Demand and Without Service Delay Supply of Energy to End Users," *Indian Patent Filed*, File No. 201631007632, Date March 4, 2016.

## 8. Referee Services:

- IEEE Journal on Selected Areas in Communications (JSAC)
- IEEE Transactions on Network and Service Management (TNSM)
- IEEE Transactions on Communications (TCOM)
- IEEE Transactions on Vehicular Technology (TVT)

- IEEE Transactions on Mobile Computing (TMC)
- IEEE Transactions on Smart Grid (TSG)
- IEEE Transactions on Sustainable Energy (TSST)
- IEEE Transactions on Sustainable Computing (TSUSC)
- IEEE Systems Journal
- IEEE Access
- IEEE IoT Journal
- IEEE Communications Magazine
- Pervasive and Mobile Computing Journal (PMC) (Elsevier)
- IET Networks
- IET Generation
- IET Wireless Sensor Systems
- Transmission & Distribution
- International Journal of Communication Systems (Wiley)
- Wireless Communications and Mobile Computing (WCMC) (Wiley)
- International Journal of Communication Networks and Distributed Systems (Springer)
- IEEE International Conference on Communications (ICC) 2019, 2020
- IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS) 2016
- IEEE Students' Technology Symposium (TechSym) 2016
- IEEE TechSym 2014

#### 9. Awards:

- Received the **HoT Topic Paper Award** in the 2<sup>nd</sup> Secure and Dependable Software Defined Networking for Sustainable Smart Communities (SecSDN) in conjunction with IEEE International Conference on Communications (ICC) 2019 for the paper entitled "D2M: Mobility-Aware Dynamic Data Multicasting in Software-Defined Data Center Networks".
- Received the **Tata Consultancy Services (TCS) Research Fellowship** (2015-2019).
- Received the **Institute** (Indian Institute of Technology Kharagpur) **Full Financial Grant** for the 6<sup>th</sup> IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, December 2014.
- Received the **Institute** (Indian Institute of Technology Kharagpur) **Full Financial Grant** for the 11<sup>th</sup> IEEE India Conference on Emerging Trends and Innovation in Technology (INDICON), Pune, India, December 2014.

- **Won** the *Best Poster Presentation Award* on the 6<sup>th</sup> Research Scholars' Day 2015 of School of Information Technology, Indian Institute of Technology Kharagpur.
- Received the **Merit scholarship** by Government of India Ministry of Human Resource Development Department of Higher Education based on performance in Higher Secondary Examination.
- Received the **Merit scholarship** by Government of India Ministry of Human Resource Development Department of Higher Education for exemplary performance in Secondary Examination.

#### 10. Achievements:

- **Served** as *Departmental Research Scholar Representative* of School of Information Technology, Indian Institute of Technology Kharagpur for academic year 2015-2016.
- **Served** as **Organizing team member** of *AICTE/QIP Sponsored Short-term course* on "Internet of Things: Convergence of Sensing, Cloud and Big-Data Networking" at Indian Institute of Technology Kharagpur, July 2015.
- **Served** as **Organizing team member** of *International Summer and Winter Term (ISWT) course* on "Enabling Internet of Things with Cloud and Big Data Networking" at Indian Institute of Technology Kharagpur, June 2015.
- **Served** as *Departmental Research Scholar Representative* of School of Information Technology, Indian Institute of Technology Kharagpur for academic year 2014-2015.
- **Elected** *Student Senate Member (SSM)* and *Vice-President (VP)* of Vikram Sarabhai Residential Complex, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India for academic year 2014-2015.
- **Ranked** 3<sup>rd</sup> and 49<sup>th</sup> in *Secondary Examination* from District - North 24 Parganas, West Bengal, India and State - West Bengal, India, respectively.

#### 11. Professional Affiliations

- Student member, ACM
- Student member, IEEE
- Student member, IEEE Young Professionals
- Student member, IEEE Communications Society
- Student member, IEEE Computer Society
- Student member, IEEE Information Theory Society