# Shanu_Thesis.pdf

Delhi Technological University

## Document Details

Submission ID

**trn:oid:::27535:121158256**

Submission Date

**Nov 12, 2025, 11:09 PM GMT+5:30**

Download Date

**Nov 12, 2025, 11:15 PM GMT+5:30**

File Name

**Shanu_Thesis.pdf**

File Size

**7.1 MB**

**111 Pages**

**35,111 Words**

**209,705 Characters**

# 9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 8 words)

## Exclusions

- 2 Excluded Sources

## Match Groups

**262**Not Cited or Quoted 9%
Matches with neither in-text citation nor quotation marks

**0** Missing Quotations 0%
Matches that are still very similar to source material

**0** Missing Citation 0%
Matches that have quotation marks, but no in-text citation

**0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

## Top Sources

| | |
|---|---|
| 7% | Internet sources |
| 6% | Publications |
| 8% | Submitted works (Student Papers) |

## Integrity Flags

**0 Integrity Flags for Review**

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

## Match Groups

**262**Not Cited or Quoted 9%
Matches with neither in-text citation nor quotation marks

**0** Missing Quotations 0%
Matches that are still very similar to source material

**0** Missing Citation 0%
Matches that have quotation marks, but no in-text citation

**0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

## Top Sources

| | |
|---|---|
| 7% | Internet sources |
| 6% | Publications |
| 8% | Submitted works (Student Papers) |

## Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

| | | | |
|---|---|---|---|
| 1 | Internet | link.springer.com | <1% |
| 2 | Internet | dspace.dtu.ac.in:8080 | <1% |
| 3 | Submitted works | American National University on 2025-05-17 | <1% |
| 4 | Internet | assets-eu.researchsquare.com | <1% |
| 5 | Submitted works | dtusimilarity on 2024-05-29 | <1% |
| 6 | Internet | arxiv.org | <1% |
| 7 | Submitted works | University Politehnica of Bucharest on 2024-11-14 | <1% |
| 8 | Internet | assets.researchsquare.com | <1% |
| 9 | Submitted works | Delhi Technological University on 2017-02-07 | <1% |
| 10 | Internet | edu.anarcho-copy.org | <1% |

| | | | |
|---|---|---|---|
| 11 | Internet | www.mdpi.com | <1% |
| 12 | Submitted works | Cranfield University on 2025-11-04 | <1% |
| 13 | Internet | bdtd-dev.sc.usp.br | <1% |
| 14 | Submitted works | AUT University on 2025-06-05 | <1% |
| 15 | Submitted works | Sheffield Hallam University on 2021-01-14 | <1% |
| 16 | Submitted works | National College of Ireland on 2020-04-23 | <1% |
| 17 | Submitted works | Strategic Education on 2025-04-28 | <1% |
| 18 | Publication | Muhammad Nadeem, Chen Hongsong. "Protecting social networks against Dual-... | <1% |
| 19 | Submitted works | University of Portsmouth on 2022-05-18 | <1% |
| 20 | Internet | www.frontiersin.org | <1% |
| 21 | Internet | nano-ntp.com | <1% |
| 22 | Internet | jowua.com | <1% |
| 23 | Publication | Lilima Jain, Venkanna U., Satyanarayana Vollala. "FTSheild: An intelligent framew... | <1% |
| 24 | Internet | ouci.dntb.gov.ua | <1% |

| | | | |
|---|---|---|---|
| 25 | Internet | www.igi-global.com | <1% |
| 26 | Publication | Okacha Amraouy, Yassine Boukhali, Aziz Bouazi, Mohammed Nabil Kabbaj, Moha... | <1% |
| 27 | Internet | ir.unimas.my | <1% |
| 28 | Internet | ejournals.itda.ac.id | <1% |
| 29 | Internet | acikbilim.yok.gov.tr | <1% |
| 30 | Submitted works | Sydney Institute of Technology and Commerce on 2025-01-12 | <1% |
| 31 | Internet | kuey.net | <1% |
| 32 | Submitted works | Mustansiriyah University on 2025-10-21 | <1% |
| 33 | Publication | "Intelligent Systems, Blockchain, and Communication Technologies", Springer Sci... | <1% |
| 34 | Submitted works | universititeknologimara on 2025-03-14 | <1% |
| 35 | Internet | www.informatica.si | <1% |
| 36 | Submitted works | Tarleton State University on 2025-05-01 | <1% |
| 37 | Internet | nijocet.fud.edu.ng | <1% |
| 38 | Submitted works | University of Technology, Sydney on 2024-09-20 | <1% |

| | | | |
|---|---|---|---|
| 39 | Internet | peerj.com | <1% |
| 40 | Submitted works | University of Greenwich on 2025-09-08 | <1% |
| 41 | Internet | download.bibis.ir | <1% |
| 42 | Internet | repositorio.pucrs.br | <1% |
| 43 | Publication | "Convergence of AI, Federated Learning, and Blockchain for Sustainable Develop... | <1% |
| 44 | Publication | Samarjeet Borah, Ratna Raja Kumar Jambi, Sharifah Sakinah Syed Ahmad, Mahen... | <1% |
| 45 | Submitted works | University of Greenwich on 2024-09-05 | <1% |
| 46 | Submitted works | Brunel University on 2023-09-01 | <1% |
| 47 | Submitted works | Delhi Technological University on 2025-05-06 | <1% |
| 48 | Submitted works | Middlesex University on 2024-09-25 | <1% |
| 49 | Publication | Shafiullah Khan, Al-Sakib Khan Pathan, Nabil Ali Alrajeh. "Wireless Sensor Networ... | <1% |
| 50 | Submitted works | Sir Syed University of Engineering and Technology, Karachi on 2025-10-14 | <1% |
| 51 | Submitted works | Universiti Sains Malaysia on 2023-10-25 | <1% |
| 52 | Publication | Ye, Jingting. "ResSDN: A Machine Learning-Based Tool for Joint QoS-Security Opti... | <1% |

| | | | |
|---|---|---|---|
| 53 | Internet | cs.adelaide.edu.au | <1% |
| 54 | Internet | turcomat.org | <1% |
| 55 | Submitted works | Anna University on 2024-12-28 | <1% |
| 56 | Submitted works | Birla Institute of Technology and Science Pilani on 2017-11-06 | <1% |
| 57 | Submitted works | Brunel University on 2023-09-01 | <1% |
| 58 | Submitted works | Florida Institute of Technology on 2023-04-22 | <1% |
| 59 | Submitted works | Intercollege on 2023-04-09 | <1% |
| 60 | Submitted works | Arts, Sciences & Technology University In Lebanon on 2024-01-20 | <1% |
| 61 | Submitted works | Fakultet elektrotehnike i računarstva / Faculty of Electrical Engineering and Com... | <1% |
| 62 | Publication | Liehuang Zhu, Md M. Karim, Kashif Sharif, Chang Xu, Fan Li, Xiaojiang Du, Mohse... | <1% |
| 63 | Submitted works | Napier University on 2015-08-17 | <1% |
| 64 | Submitted works | University of Maryland, Global Campus on 2025-08-12 | <1% |
| 65 | Internet | acperpro.com | <1% |
| 66 | Internet | core.ac.uk | <1% |

| | | | |
|---|---|---|---|
| 67 | Internet | opus.lib.uts.edu.au | <1% |
| 68 | Internet | sciencetransactions.com | <1% |
| 69 | Internet | www.researchgate.net | <1% |
| 70 | Submitted works | Middlesex University on 2024-04-14 | <1% |
| 71 | Publication | Yi Shen, Chunming Wu, Qiumei Cheng, Dezhang Kong. "AFTM: An Adaptive Flow T... | <1% |
| 72 | Internet | www.slideshare.net | <1% |
| 73 | Internet | www.wjgnet.com | <1% |
| 74 | Publication | Youssef Baddi, Yassine Maleh, Izzat Alsmadi, Mohammed Lahby. "Generative AI f... | <1% |
| 75 | Internet | mural.maynoothuniversity.ie | <1% |
| 76 | Internet | qspace.qu.edu.qa | <1% |
| 77 | Internet | technodocbox.com | <1% |
| 78 | Internet | www.ijcopi.org | <1% |
| 79 | Internet | 123dok.net | <1% |
| 80 | Submitted works | University of Waikato on 2023-11-27 | <1% |

81 Internet

dtu.ac.in <1%

82 Internet

ir.upsi.edu.my <1%

83 Publication

Harris, Douglas O., Jr.. "Navigating Threats: Intelligent Models for Advanced Intru... <1%

84 Internet

res.ijsrcseit.com <1%

85 Publication

Carlos Núñez-Gómez, Carmen Carrión, Blanca Caminero, Francisco M. Delicado. "... <1%

86 Submitted works

Colorado Technical University Online on 2014-06-20 <1%

87 Publication

Saqer, Khansa. "Blockchain-Based Approach for Enhancing Resilience Against DD... <1%

88 Submitted works

University of Surrey Roehampton on 2011-08-19 <1%

89 Internet

conferencealerts.com <1%

90 Internet

scholar.uoc.ac.in <1%

91 Submitted works

Bahrain Polytechnic on 2024-06-17 <1%

92 Submitted works

Bahria University, Islamabad on 2025-11-12 <1%

93 Submitted works

Indian Institute of Information Technology, Nagpur on 2025-10-27 <1%

94 Submitted works

Liverpool John Moores University on 2024-12-18 <1%

| | | |
|---|---|---|
| 95 | Publication | |
| Maedeh Abedini Bagha, Kambiz Majidzadeh, Mohammad Masdari, Yousef Farhan... | | <1% |
| 96 | Publication | |
| Nobuhide Nonaka, Kazushi Muraoka, Satoshi Suyama, Jun Mashino et al. "Indoor ... | | <1% |
| 97 | Submitted works | |
| UNICAF on 2024-12-08 | | <1% |
| 98 | Submitted works | |
| University College London on 2014-09-01 | | <1% |
| 99 | Submitted works | |
| University of Anbar on 2023-05-23 | | <1% |
| 100 | Submitted works | |
| University of Hertfordshire on 2025-09-04 | | <1% |
| 101 | Internet | |
| hsnlab.tmit.bme.hu | | <1% |
| 102 | Internet | |
| jyx.jyu.fi | | <1% |
| 103 | Internet | |
| singipedia.singidunum.ac.rs | | <1% |
| 104 | Internet | |
| www3.nd.edu | | <1% |
| 105 | Submitted works | |
| Bergen Community College on 2024-12-02 | | <1% |
| 106 | Submitted works | |
| Chester College of Higher Education on 2024-02-14 | | <1% |
| 107 | Submitted works | |
| Cranfield University on 2012-06-11 | | <1% |
| 108 | Publication | |
| Kokila M Kokila M, Srinivasa Reddy Konda Srinivasa Reddy Konda. "DeepSDN: Dee... | | <1% |

109 Submitted works

Macquarie University on 2024-10-30 <1%

110 Submitted works

Middlesex University on 2018-12-14 <1%

111 Submitted works

Middlesex University on 2024-09-27 <1%

112 Submitted works

Staffordshire University on 2024-05-12 <1%

113 Submitted works

University of Bristol on 2014-11-07 <1%

114 Submitted works

University of Central Lancashire on 2010-05-21 <1%

115 Submitted works

University of Greenwich on 2023-03-24 <1%

116 Submitted works

University of Sunderland on 2023-03-06 <1%

117 Internet

dspace.daffodilvarsity.edu.bd:8080 <1%

118 Internet

kuk.ac.in <1%

119 Internet

webthesis.biblio.polito.it <1%

120 Internet

www.exabeam.com <1%

121 Publication

"Fifth International Conference on Computing and Network Communications", S... <1%

122 Publication

"Web, Artificial Intelligence and Network Applications", Springer Science and Busi... <1%

123 Submitted works

AUT University on 2025-10-27 <1%

124 Publication

Abdelhamied A. Ateya, Ammar Muthanna, Anastasia Vybornova, Abeer D. Algarni... <1%

125 Publication

Abhishek Narwaria, Arka Prokash Mazumdar. "Software-Defined Wireless Sensor ... <1%

126 Publication

Akhilesh Kumar Yadav. "Computational Techniques in Environmental Engineerin... <1%

127 Publication

Al-Sakib Khan Pathan, Muhammad Mostafa Monowar, Zubair Md. Fadlullah. "Buil... <1%

128 Publication

Alshaya, Abdullah. "Software-Defined Networking Security Techniques and the Di... <1%

129 Submitted works

Anna University on 2025-01-23 <1%

130 Submitted works

Army Institute of Technology on 2025-10-14 <1%

131 Submitted works

Arts, Sciences & Technology University In Lebanon on 2025-01-21 <1%

132 Publication

Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artific... <1%

133 Submitted works

Bannari Amman Institute of Technology on 2024-08-20 <1%

134 Submitted works

Capella University on 2024-11-11 <1%

135 Publication

Chen, Mingming. "Advancing Network Security in the Era of Network Programma... <1%

136 Publication

Elmeleegy, Khaled. "Enhancing ethernet's reliability and scalability", Proquest, 20... <1%

137 Publication

Emmanuel, Ikiomoye Douglas. "Proposed New SD-WAN Architecture to Facilitate ... <1%

138 Submitted works

Fachhochschule Kärnten Gemeinnützige Privatstiftung on 2024-09-12 <1%

139 Publication

Fei Hu. "Network Innovation through OpenFlow and SDN - Principles and Design"... <1%

140 Submitted works

Global Banking Training on 2025-07-24 <1%

141 Submitted works

Glyndwr University on 2025-01-12 <1%

142 Publication

Ishfaq Ahmad, Sanjay Ranka. "Handbook of Energy-Aware and Green Computing,... <1%

143 Submitted works

Jose Rizal University on 2024-12-03 <1%

144 Publication

Kaixi Wang, Yunhe Cui, Guowei Shen, Chun Guo, Yi Chen, Qing Qian. "PRAETOR:Pa... <1%

145 Publication

Leila Shirani, Homa Movahednejad, Mahdi Sharifi, Marjan Mahmoudi. "Improving... <1%

146 Publication

Marcos V.O. Assis, Luiz F. Carvalho, Jaime Lloret, Mario L. Proença. "A GRU deep le... <1%

147 Submitted works

Middlesex University on 2017-04-18 <1%

148 Submitted works

Mirpur University of Science & Technology, AJK on 2025-07-17 <1%

149 Publication

Mitali Sinha. "A Comprehensive Survey of DDoS Attack Defense Systems for Differ... <1%

150 Publication

Mosab Hamdan, Entisar Hassan, Ahmed Abdelaziz, Abdallah Elhigazi et al. "A com... <1%

| | | |
|---|---|---|
| 151 | Publication | |
| Natalia C. Fernandes, Marcelo D. D. Moreira, Igor M. Moraes, Lyno Henrique G. Fe... | | <1% |
| 152 | Submitted works | |
| RMIT University on 2025-10-14 | | <1% |
| 153 | Publication | |
| Rashmi Gupta, Xiomara Patricia Blanco Valencia, Lalit Mohan Goyal, Jeetendra Ku... | | <1% |
| 154 | Publication | |
| Silva, Henrique Miguel Simões. "Fairness Centred Framework for the ONOS SDN C... | | <1% |
| 155 | Submitted works | |
| Stefan cel Mare University of Suceava on 2011-11-02 | | <1% |
| 156 | Publication | |
| Talluri, Ashmitha. "Design, Implementation, and Analysis of a Low-Power Success... | | <1% |
| 157 | Submitted works | |
| Tampere University on 2025-11-01 | | <1% |
| 158 | Submitted works | |
| The Independent Institute of Education (IIE) on 2025-10-13 | | <1% |
| 159 | Submitted works | |
| University of Greenwich on 2025-05-01 | | <1% |
| 160 | Submitted works | |
| University of Hertfordshire on 2024-07-05 | | <1% |
| 161 | Submitted works | |
| University of Houston System on 2024-11-28 | | <1% |
| 162 | Submitted works | |
| University of Wales, Lampeter on 2024-08-26 | | <1% |
| 163 | Submitted works | |
| University of Warwick on 2024-09-02 | | <1% |
| 164 | Submitted works | |
| University of West London on 2016-01-09 | | <1% |

| | | | |
|---|---|---|---|
| 165 | Internet | c.coek.info | <1% |
| 166 | Internet | cdn.iiit.ac.in | <1% |
| 167 | Internet | dns1.mnit.ac.in | <1% |
| 168 | Internet | docplayer.net | <1% |
| 169 | Internet | dokumen.pub | <1% |
| 170 | Internet | ebin.pub | <1% |
| 171 | Internet | eprints.glos.ac.uk | <1% |
| 172 | Internet | iariajournals.org | <1% |
| 173 | Internet | ijetcse.com | <1% |
| 174 | Internet | informatica.si | <1% |
| 175 | Internet | ir.knust.edu.gh | <1% |
| 176 | Internet | nordopen.nord.no | <1% |
| 177 | Internet | osuva.uwasa.fi | <1% |
| 178 | Internet | publications.eai.eu | <1% |

| | | | |
|---|---|---|---|
| 179 | Internet | sites.ieee.org | <1% |
| 180 | Internet | tmonnier.com | <1% |
| 181 | Internet | uhbvn.org.in | <1% |
| 182 | Internet | vuir.vu.edu.au | <1% |
| 183 | Internet | www.atlantis-press.com | <1% |
| 184 | Internet | www.diva-portal.org | <1% |
| 185 | Internet | www.tdx.cat | <1% |
| 186 | Internet | www2.mdpi.com | <1% |

# A NOVEL FRAMEWORK FOR THE NETWORK TRAFFIC ANALYSIS USING A 42 CONTROLLER IN SOFTWARE-DEFINED NETWORKING

**A Thesis Submitted**
**In Partial Fulfillment of the Requirements for the Degree of**

## DOCTOR OF PHILOSOPHY

**in**

**Computer Science & Engineering**

**by**

**Shanu Bhardwaj**
2 **(2K21/PHDCO/01)**

**Under the Supervision of**

118 **Prof. Shailender Kumar**
**(Supervisor)**
Department of Computer
Science & Engineering
Delhi Technological University

**Dr. Ashish Girdhar**
**(Co-Supervisor)**
Department of Computer
Science & Applications
Kurukshetra University

5 **Department of Computer Science & Engineering**

**DELHI TECHNOLOGICAL UNIVERSITY**
**(Formerly Delhi College of Engineering)**
**Shahbad Daulatpur, Main Bawana Road, Delhi-110042, India**

**December, 2025**

# ACKNOWLEDGEMENT

I express my profound gratitude to Almighty God for providing me the strength, resilience, and guidance to pursue and complete this research journey. I am deeply indebted to my supervisors, Prof. Shailender Kumar and Dr. Ashish Girdhar, for their invaluable mentorship, constant encouragement, and insightful suggestions throughout this journey. Prof. Shailender Kumar's technical expertise and thoughtful guidance have been instrumental in overcoming the challenges faced during my research. Dr. Ashish Girdhar has been a constant source of motivation and support. His leadership and vision inspired me to strive for excellence. Also, my sincere thanks to Prof. Manoj Kumar, HOD, (Dept. of Computer Science and Engineering) for insightful comments and valuable suggestions. I extend my heartfelt thanks to the esteemed faculty members of the Department of Computer Science and Engineering for their unwavering support and encouragement. Their advice and collaborative spirit have enriched my academic experience and contributed significantly to my personal and professional growth.

I would also like to acknowledge the continuous support and encouragement provided by Prof. Prateek Sharma, Vice-Chancellor. His dedication to fostering a research-oriented environment has been a significant driving force behind my accomplishments.

Finally, with a heart full of love and longing, I offer my deepest gratitude to supreme Supervisor my Mother, Mrs. Nitu Bhardwaj, my Father, Mr. Sanjay Bhardwaj, my Brother, Mr. Saksham Bhardwaj,and my Husband, Mr. Ravi Deswal. A very special thanks to my dearest daughter Adrija Deswal Bhardwaj. Their values and teachings remain my guiding light, and gives me strength and purpose on this journey. This acknowledgment is a humble testament to the collective efforts and support of all these individuals, whose contributions have been pivotal to the successful completion of my doctoral research.

**Shanu Bhardwaj**
**Roll No. 2K21/PHDCO/01**

# DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

## CANDIDATE DECLARATION

I, Shanu Bhardwaj (2K21/PHDCO/01), Research Scholar in the Department of Computer Science & Engineering, hereby declare that the work which is being presented in the thesis entitled "A Novel framework for the Network Traffic Analysis using a controller in Software-Defined Networking" in partial fulfillment of the requirements for the award of the Degree of Doctor of Philosophy, submitted in the Department of Computer Science & Engineering, Delhi Technological University is an authentic record of my own work carried out during the period from August, 2021 to December, 2025 under the supervision of Prof. Shailender Kumar (Supervisor) and Dr. Ashish Girdhar (Co- Supervisor) of Department of Computer Science and Engineering, Delhi Technological University, Delhi, India. The matter presented in the thesis has not been submitted by me for the award of any other degree of this or any other Institute.

**Shanu Bhardwaj**

Place: New Delhi

Date: 31-10-2025

2 **DELHI TECHNOLOGICAL UNIVERSITY**
(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

## SUPERVISOR(s) CERTIFICATE

This is to certify that the work embodied in this thesis entitled **“A Novel framework for the Network Traffic Analysis using a controller in Software-Defined Networking”** done by Shanu Bhardwaj, roll no. 2K21/PHDCO/01 in the Department 9 of Computer Science & Engineering, Delhi Technological University is an authentic work carried out by him under our guidance.

This work is based on original research and the matter embodied in this thesis has not been submitted earlier for the award of any degree or diploma to the best of our knowledge and belief.

**Prof. Shailender Kumar**
Professor
54 Department of Computer
Science & Engineering
Delhi Technological University
Delhi, India

**Dr. Ashish Girdhar**
Assistant Professor
Department of Computer
Science & Applications
Kurukshetra University
Kurukshetra, India

Date: 31-10-2025

# ABSTRACT

The rapid growth of modern networks and diverse traffic patterns have highlighted traffic management as a core challenge in network administration. Traditional networks, with their rigid architectures and limited programmability, fail to meet the dynamic requirements of today's applications. Software Defined Networking (SDN) has emerged as a novel paradigm that decouples the control and data planes, enabling centralized control and intelligent network programmability. This thesis outlines a topology-aware intelligent network traffic analysis framework using Ryu SDN controller for enhanced network performance and decision-making efficiency.

A topology-aware SDN environment is designed using Mininet as the emulator and OpenFlow as the communication protocol. The proposed framework leverages the Ryu controller's Python-based modular architecture to implement dynamic traffic analysis and adaptive flow management. Various network topologies are constructed to simulate diverse operational environments and evaluate the framework's adaptability. The described SDN environment enables real-time monitoring of network parameters and flow optimization, ensuring effective data transfer under various traffic loads.

Performance evaluation is conducted using key parameters, including latency, throughput, jitter, packet loss, and controller response time, across different network conditions. The obtained results indeed present a significant enhancement in network performance, as they generate up to a 22% gain in throughput and a 25% reduction in latency, along with decreased packet loss. Importantly, the comparative benchmarking confirms the performance robustness and scalability of the proposed SDN model, especially for more dynamic and larger topologies.

As a result, this research contributes to the advancement of SDN-based network intelligence by combining topology awareness alongside traffic analysis and performance monitoring. The implications of this work lay the foundation for deploying efficient, scalable, and adaptable network management solutions applicable to real-world domains, such as cloud computing, and IoT-driven systems.

# LIST OF PUBLICATIONS

| **Research Objectives** | **Publications** |
|---|---|
| **RO1: To investigate the existing network traffic performance analysis in the SDN controller.** | **Conference: Published** article entitled "Software defined Networking: A Traffic Engineering Approach" in ***IEEE 8th International Conference on Electrical, Electronics and Computer Engineering, UPCON, 2021.*** **DOI: 10.1109/UPCON52273.2021.9667584 (SCOPUS Indexed)** |
| **RO2: To develop the network topology for traffic analysis using SDN controller.** | **Journal: Published** article entitled "Network Traffic Analysis in Software-Defined Networking Using RYU Controller" in ***Wireless Personal Communications*****, 2023. DOI: https://doi.org/10.1007/s11277-023-10680-1 (SCIE, IF-1.9)**<br>**Journal: Accepted article entitled "**Performance Evaluation of SDN Controllers: Analysing the TCP traffic management in POX, Ryu, and ODL**"** *in* ***International Journal of Advances in Soft Computing and Intelligent Systems (IJASCIS)*** *2024, Vol 03, Issue 02, 303-314 ISSN: 3048-4987* **(SCOPUS Indexed, H-index-26)** |
| **RO3: To propose a framework and analyze the performance of the developed network topology using Ryu controller.** | **Conference: Published** article "Current Perspectives and Virtualization Solutions with SDN for IoT" in ***International Conference on Smart Technologies for Smart Nation (SmartTechCon, 2023), Singapore.*** **DOI: 10.1109/SmartTechCon57526.2023.10391403 (SCOPUS Indexed)**<br>**Journal:** Communicated article entitled "Performance evaluation of RYU controller in SDN using shortest path algorithm" **in Wireless Personal Communications.** ***(Minor Revision)***<br>**Journal:** Communicated article entitled "IntelliSDN: A Unified Framework for Intelligent Traffic Analysis in Software-Defined Networks" in **Cluster Computing. (Under Review)** |
| **RO4: To evaluate the performance of the proposed framework based on execution parameters and perform a comparative analysis with the existing framework.** | **Journal: Published** article entitled "A Networking Framework to Analyse the Performance of TEVN Using Ryu Controller for Network Optimization in SDN" **in *Journal of Information Science and Engineering.* DOI: 10.6688/JISE.202601_42(1).0003 (SCIE, IF-1.42)**<br>**Journal:** Communicated article entitled "A framework to develop realistic virtual network for traffic analysis using Ryu controller in SDN" in ***Wireless Networks. (under review)*** |

# TABLE OF CONTENTS

117

# LIST OF TABLES

77

# LIST OF FIGURES

50

41

119

# LIST OF ABBREVIATIONS

| | |
|---|---|
| IoT | Internet of Things |
| SDN | Software-Defined Networking |
| NPC | Network Processor Cards |
| ACL | Access Control List |
| IDS | Intrusion Detection System |
| DDoS | Distributed Denial of Service |
| QoS | Quality of Service |
| NFV | Network Functions Virtualization |
| TCP | Transmission Control Protocol |
| ONOS | Open Network Operating System |
| DPI | Deep Packet Inspection |
| SNMP | Simple Network Management Protocol |
| ODL | OpenDaylight controller |
| API | Application Programming Interface |
| NBI | Northbound Interface |
| CDPI | Control Panel to Drive Interface |
| PDR | Packet Delivery Ratio |
| ICMP | Internet Control Message Protocol |
| RTT | Round Trip Time |
| ML | Machine Learning |
| AI | Artificial Intelligence |

# CHAPTER 1

# INTRODUCTION

In several sectors, including education, healthcare, banking, e-commerce, and defense systems, among others, computer networks are now the primary means through which people engage in communication, share information, or receive services [1]. There is more going on now than mere information sharing. They can also assist with new technologies that must be fast, predictable, and safe, while promoting safe teamwork and real-time communication. But decades into an era of technological advancement, old networking architectures are still struggling to keep pace with the complexity new apps and services bring.

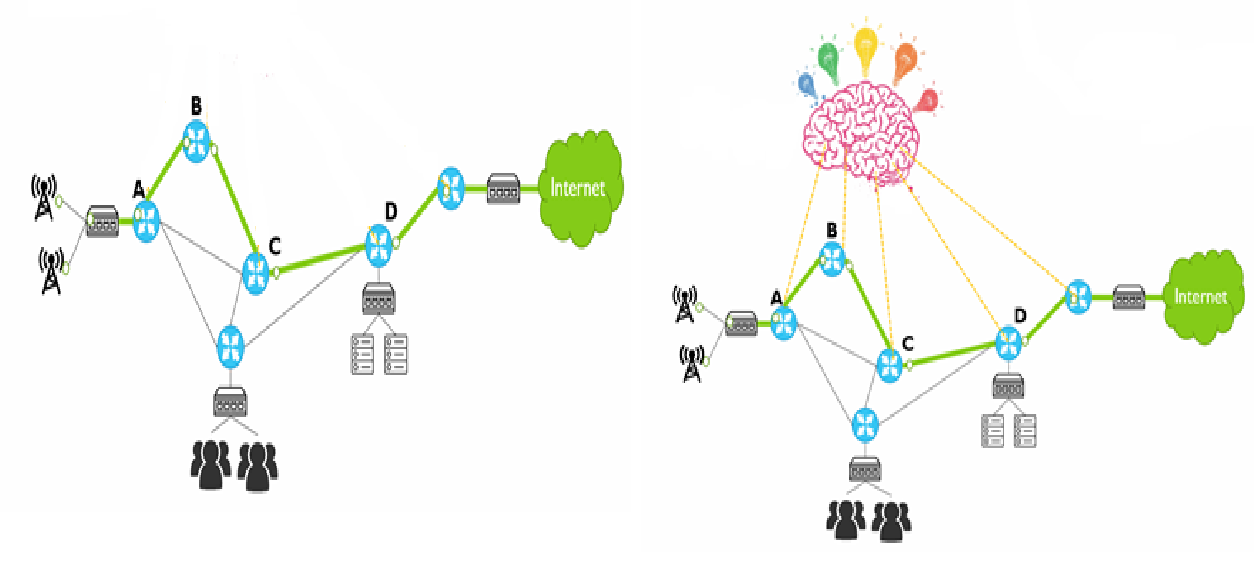
For conventional networks, separation between the control plane and the data plane might not be strict at all. This implies that routers, switches, and even firewalls can operate independently and maintain/employ forwarding entries locally [2]. This model has been the standard for a long time, but it has numerous problems. Device-level management can be a chore when we have many of them, because there is a need to configure and monitor each one individually. New nodes are added or traffic policies are modified manually, and therefore, they take considerable time to scale. Additionally, vendor-specific implementations also lock companies into solutions that are costly and difficult to change, due to their reliance on hardware. Furthermore, traditional networks are not adaptable to the real-time shifts that dynamic workloads necessitate. It is not a very robust technology, which opens doors to numerous vulnerabilities, detrimental to the current world of cloud computing, the Internet of Things, 5G services, and apps that require low latency [3]. The difference between traditional networking and SDN is illustrated in Figure 1.1.

The surge of IoT devices, edge computing, cloud platforms, and fast multimedia services has only exacerbated the issues with traditional networks. For example, IoT solutions can support billions of devices exchanging small but frequent data flows, which pose challenges that no static, rule-based architecture can overcome. Likewise, applications such as self-driving cars and telemedicine, which 5G enables, have extremely low latency requirements and require bandwidth to be allocated on the fly, a capability that older systems cannot achieve very well. And this is what has allowed even SDN a civilizational reboot in how we build things.

In short, it separates the decision-making mechanism and the packet forwarding mechanism. Instead of letting every individual device make decisions on its own, SDN centralizes the network intelligence in a software-based controller. The hardware passes the data. Numerous advantages accompany this significant change.

Providing a global perspective on the network enables administrators to dynamically redefine resources, automate configurations, and enforce policies uniformly across the organization. It is also well-suited for fast adaptation as the controller can instantly respond to changes in traffic flows. With SDN, you gain the scalability, flexibility, and automation that traditional networks lack. [4]

One of the key features of SDN is its ability to monitor network traffic in real-time. Having a centralized view of the entire network enables deep traffic insights, allows for the analysis of flows, and facilitates troubleshooting while enforcing rigorous security policies. For instance, bandwidth can be dynamically reserved for critical applications, and packets that appear suspicious can be rerouted or dropped. This is why SDN in the enterprise data center, financial platform, or defense network becomes particularly appealing. Due to imperfect obliviousness, traffic analysis remains a significant concern in SDN [5]. Scalability remains a primary concern, as the controller can become a bottleneck when large amounts of traffic are present. Latency in analysis and resolution decreases responsiveness, which is further complicated by IoMT or custom topology that introduces varied traffic patterns. Moreover, much of the recent research concentrates on SDN behavior in general and does not cover traffic performance analysis for custom or complex networks, such as [6]. These issues suggest the necessity for new frameworks to achieve efficient, reliable, and scalable traffic analysis under current scenarios. One scalability issue is that the controller can become a bottleneck when traffic volumes are high in Switches with Network Processor Cards (NPCs). In this section, we demonstrate how selective replication alleviates the processing overhead of switches equipped with network processor cards.



(a) (b)

Figure 1.1: Overview of (a) Traditional and (b) SDN Networking

## 1.1 Background

11 Network traffic analysis is a critical component of network security and management; however, in the modern world, understanding network traffic is more important than ever. Networks are further complicated as an increasing flow of information is created, the speed at which cloud-based applications are adopted and available on a network, and the number of mobile devices and IoT endpoints organizations use continues to grow [7]. Now, traditional methods of traffic monitoring, such as NetFlow, passive packet sniffers, and event-driven rule bases for firewalls, were not created in smaller, more stable settings. Although these methods were plausible in the past, none of them can meet the velocity and variability of traffic in the present and future times in a functional manner. SDN offers a paradigm change to solve these challenges.

In traditional architectures, every device on the network is independent. With SDN, administrators have centrally controlled access to the entire network via a logically centralized controller [8]. Transitioning to a centralized approach enables central traffic analysis and policy enforcement, with scalability and velocity that are not attainable under legacy systems.

### 1.1.1 Traditional Network Limitations

64 Routers and switches are examples of legacy networking gear that possess both a data plane and a control plane, as shown in Figure 1.2. The device uses its own rules to decide how to send packets. This design has many problems:

- Static Configurations: If a DDoS attack hits unexpectedly, we would need to reconfigure each router ourselves.
- Vendor Lock-in: As an example, a Cisco router could employ management protocols that are only accessible to Cisco devices, and it would be challenging to manage Juniper or Huawei devices. This complicates the use of multiple vendors in the same deployment.
- Complex Management: Modifying ACLs by hand on thousands of switches in a big business network can take hours, which gives attackers time to take advantage of the situation.
- Limited Responsiveness: When there is a sudden spike in video traffic during live streaming events, the network can't handle it dynamically, which causes congestion and lower QoS.

### 1.1.2 Integration with NFV

64 110 Routers and switches are some examples of legacy networking gear that possess both a Data Plane and a control Plane [9]. The device uses its own rules to decide how to send packets. This design has many problems:

- Elastic Scaling: For example, during an online shopping event like 'Black Friday', when traffic is high, additional virtual firewalls or load balancers can be added on demand to absorb the extra load.
- On-Demand Deployment: For example, in response to malware traffic at a particular edge node, an IDS can be deployed at that edge node in the next moments.

- Resource Efficiency: Previously, each appliance would need to be a hardware appliance that incurred significant capital and operational costs, but now virtualized functions can run on inexpensive servers instead.
- Scope of Infrastructural and Cost Efficiency: Along with SDN, NFV provides additional efficiencies, as the virtualized functions that we are running can run on less expensive servers, and instead of being standalone hardware appliances in hundreds of locations, you can orchestrate them via SDN/NFV.

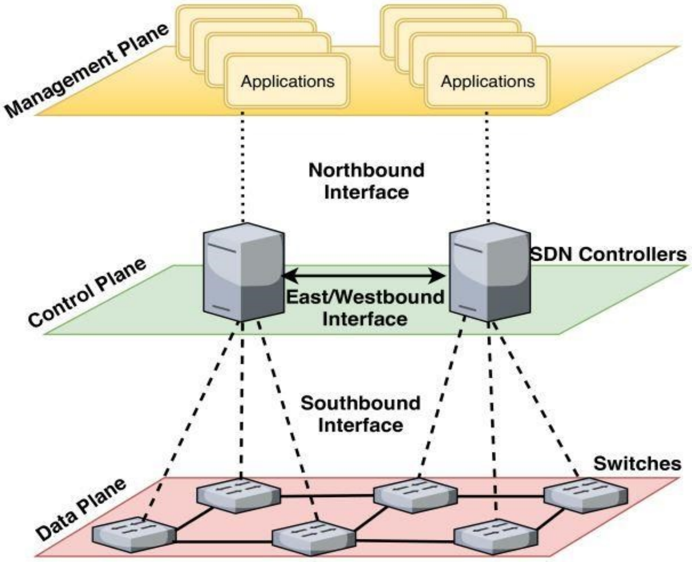


Figure 1.2: Layer-based architecture of SDN

## 1.2 Research Challenges in SDN for Network Traffic Analysis

SDN opens up new ways to analyze traffic, but it also presents several significant problems that need to be addressed. The challenges can be divided into four categories: scalability, latency and overhead, security, and traffic heterogeneity.

### 1.2.1 Ability to Grow Challenges

Scalability is a critical issue in SDN deployments. The central controller must manage thousands and millions of flow requests simultaneously.

- Centralized Bottleneck: A centralized controller may receive millions of flow requests per second from a large data center. For instance, Facebook data centers manage terabits of traffic in one second. To manage such quantities, a single SDN controller requires access to terabytes of data; it would simply collapse the SDN controller under scale.
- High Load: Whenever any new cloud app initiates a TCP port, it makes the central controller create a new rule, which consumes enormous CPU and memory.
- Performance degradation: There is the possibility that the speed of the central controller responding to the flow request may not be quick enough once workloads escalate, because packets would not be forwarded promptly. There

would be a delay for the servers. There may also be potentially hazardous service degradation for time-protected applications such as remote surgery or online gaming.

- Due to such scalability concerns, one effective method is the use of hierarchical or distributed controllers, such as ONOS clusters.
- From a performance perspective, clusters share processing and analysis of traffic flows across distributed, disparate nodes.

### 1.2.2 Overhead and Performance Bottlenecks

154 SDN creates additional communication delays between the [network] controller and the network devices. This could make other potential latencies:

- Frequent Flow Requests: When switches continually request flow decisions from the controller, this adds the latency from the round-trip, regardless of the distance.
- Real-Time Inspection Costs: The controller frequently uses high amounts of CPU for DPI.
- A Single Point of Failure: While traditional networks have routers and switches operating independently, in SDN, if the controller goes down, traffic analysis may go down with it.

## 1.3 Research Motivation

In recent years, network traffic has grown exponentially, making modern network management and security more challenging than ever. The growth is primarily driven 116 by the increasing number of IoT devices, the rollout of 5G networks, and the growing popularity of cloud-based apps and services. Traffic was more predictable in the past, and it was possible to manage networks with static policies. However, today, due to the various and dynamic nature of digital infrastructures, these traffic patterns are large-scale and inhomogeneous. This shift places immense stress on existing monitoring and management systems, exposing their shortcomings and underscoring the need to develop new approaches.

### 1.3.1 Gaps in Current SDN-Based Solutions

SDN-based traffic analysis solutions have their advantages and limitations. If used in real-world environments, the following weaknesses should be addressed:

- Scalability: A Centralized controller cannot perform their operation effectively under very high traffic conditions.
- Overhead and Latency of Controllers: The overhead caused by periodic controller-switch communication is not tolerable for real-time applications such as high-frequency financial trading, distant robotic surgery, etc.
- Single Point of Failure: The SDN controllers make a good target because they are centralized. For example, the entire network goes down if a DDoS attack is launched at a controller.

## 1.4 Problem Definition

The rapid growth in the quantity of community site visitors and its complexity has drastically altered the way sophisticated virtual infrastructures operate. When all was well and there was equilibrium, static rule-based functions, packet inspection, flow aggregation, and other forms of traffic analysis methods provided sufficient defense for campus and enterprise networks. However, in the fast-changing environments of today, which require support for billions of devices, real-time applications, and mission-critical services, they worked by doing the job, but not enough. Digital ecosystems are very dynamic, and therefore, we need models that provide greater agility and flexibility.

### 1.4.1 Emerging Technologies Challenges

Emerging technologies, such as IoT, 5G networks, and edge computing, are also likely to disrupt traffic flow patterns compared to traditional client-server architectures.

- One aspect of IoT traffic to consider is the billions of energy-efficient devices constantly sending tiny packets. A smart city is a prime example where thousands of sensors continuously send data about the environment. Monitoring solutions designed to handle high-volume, predictable types of flows struggle to process the expected microflows efficiently.
- The second challenge is due to the requirements of 5G networks that support applications such as autonomous vehicles, AR, and telemedicine, where outlets are expected, ultra-low latency, and high reliability. Delays or issues that result in a greater analysis lead-time (even fractions of a second) to mechanisms that analyze or ensure priority across microflows can result in disastrous (or catastrophic) outcomes, such as an unwanted contact of vehicles in a vehicular network or losing patients in remote refrained surgery.
- The third challenge is due to Edge Computing, where computation takes place closer to the source of measurements and provides opportunities for distributed traffic patterns (where, after localized analysis, there is still a requirement for coordination with a significant analysis model). Traditional centralized analysis models are often less efficient than decentralized architectures.

## 1.5 Research Objectives

The primary objective of this research study is to emphasize the end-to-end quality of service in SDN-based network infrastructure, aiming to enhance resilience. During the research period, we focused on four specific objectives. The general objectives of the study are as follows:

A. To investigate the existing network traffic performance analysis in the SDN controller.

- A literature review is conducted to know about the current existing methodologies/tools used for SDN-based network traffic analysis

50
- Evaluate the performance of existing SDN controllers in terms of traffic analysis functions and identify their strengths and weaknesses.
- Examine the scalability, latency, resource consumption, and real-time processing abilities of current solutions.
- Analyze the existing traffic analysis frameworks to identify the gaps and limitations, especially regarding the modern network requirements like IoT, 5G, and edge computing.
- Investigate limitations in deploying these frameworks, including controller overhead, single points of failure, and security risks.

B. To develop the network topology for traffic analysis using an SDN controller.

- Create a representative network topology simulation of the real world with heterogeneous traffic and network link attributes.
- Use the SDN controller as a single authority that monitors the network traffic. 158
- Implement components to simulate different traffic patterns (e.g., high traffic loads, dynamic routing, and heterogeneous flows).
- Ensure the topology supports extensibility for adding new features or modules for traffic analysis.
- Integrate mechanisms to collect flow-level data and monitor network performance metrics such as throughput, delay, and packet loss. 43

C. To propose a framework and analyze the performance of the developed network topology using the Ryu controller.

- Design a novel traffic analysis framework that leverages the programmability and flexibility of the Ryu controller.
- Incorporate intelligent features, such as machine learning algorithms or anomaly detection techniques, to enhance traffic analysis capabilities.
- Optimize the framework for scalability, real-time processing, and low overhead in large and dynamic network environments.
- Deploy the framework within the developed topology to analyze and manage traffic efficiently.
- Test and fine-tune the framework's performance by simulating real-world scenarios, including high traffic volumes and security threats.

6 D. To evaluate the performance of the proposed framework based on execution parameters and perform a comparative analysis with the existing framework.

- 162 Define key performance metrics for evaluation, such as throughput, latency, resource utilization, scalability, and detection accuracy.
- 142 Conduct experiments to measure the performance of the proposed framework under varying network conditions (e.g., load variations, attacks, and dynamic routing changes).
- 122 Compare the results of the proposed framework with those of existing frameworks, highlighting improvements in performance and efficiency.
- Identify any trade-offs or limitations of the proposed framework and discuss potential solutions for overcoming them.
- Summarize the findings to demonstrate the effectiveness of the proposed framework and its contributions to SDN-based traffic analysis research.

## 1.6 Key Contribution of Research Work

7 The research presented in this thesis addresses critical challenges in SDN and Traffic analysis, offering novel solutions through comprehensive design, implementation, and evaluation. The key contributions of this work are outlined below, each reflecting a significant advancement toward achieving the research objectives. These contributions collectively highlight the originality, technical depth, and practical relevance of the proposed framework.

A. Comprehensive Review of Existing Solutions:

- Analysis of existing approaches and frameworks for network traffic analysis in an SDN-based environment.
- Identified the limitations of traditional approaches, such as scalability bottlenecks, high latency, and inadequate handling of dynamic traffic patterns.

B. Development of a Realistic Network Topology for Traffic Analysis:

- Developed and realized simulation settings that accurately reflect the real-world scenario, such as different traffic conditions and high-load situations.
- Integrated an SDN controller as the central traffic management and monitoring element for granularity over traffic analysis.

C. Proposal of a Novel Traffic Analysis Framework:

- Designed a scalable, efficient, and secure framework for network traffic analysis using the Ryu SDN controller.
- Integrated advanced features such as real-time analytics and an intelligent traffic management mechanism to address existing limitations.

D. Performance Evaluation Based on Key Metrics:

- Evaluated the suitability of architectural features in multi-dimensional network situations and in contrast with traditional measures and judgments composed of latency, throughput, scalability, resource allocation, and detection rate.
- Empirically verified hypothesis under real conditions through experimentation results (for example, higher sampled throughput under potential security risk and high load).

E. Comparative Analysis with Existing Frameworks:

- Developed and realized simulation settings that accurately reflect the real-world scenario, such as different traffic conditions and high-load situations.
- Integrated an SDN controller as the central traffic management and monitoring element for granularity over traffic analysis.

F. Advancement of SDN-Based Traffic Analysis Research:

- Contributed to the study of the SDN community on some major traffic-monitoring issues such as controller overhead, decision-making, and security loops.
- Suggested an adaptive framework that could be further customized and reused

to roll out facilitated changes to expected and new requirements and situations.

G. Integration of Emerging Technologies:

- Considered the implications of modern technologies such as IoT, 5G, and Edge computing in the design and implementation of the proposed framework.
- Ensure that the proposed model can handle time-variant and diverse traffic in the networks

## 1.7 Dissertation Organization

The thesis comprises six chapters that concisely and precisely describe the entire study. Each chapter is summarised below:

Chapter 1: Introduction

In this chapter, research is introduced by presenting some of the main concepts in Computer Networking and SDN. It traces the course of computer networking from its historical roots to the networking models we are accustomed to nowadays, based on SDN, which offers greater flexibility and programmability. The chapter also presents how traffic analysis is utilized in network management and operation, such as performance analysis, anomaly detection, and security. It discusses the motivation for analyzing traffic using SDN controllers, as a centralized approach with visibility of global information is an optimal method for making dynamic decisions. This chapter discusses existing gaps in current traffic analysis methods within an SDN environment and constructs the main problem that this research will address. It concludes by stating the research objectives, which are specific and define the boundaries of the research, and outlining the thesis content.

Chapter 2: Literature Review and Related Work

This section presents a comprehensive survey of the literature on SDN architecture, traffic analysis, research studies on topology management, and controller optimization. The chapter begins with a review of the SDN architecture and the controller's role in initiating flows. For traffic analysis, the various types of traffic analysis techniques from traditional networking and SDN are reviewed with respect to their strengths and limitations. The chapter discusses different approaches to topology design for traffic management and reviews how optimization techniques are applied to aid a controller. Furthermore, a comparative analysis is conducted on existing frameworks in SDN to benchmark applications and their outcomes. Ultimately, the comprehensive review of the related literature reveals apparent research gaps, providing a basis for proposing new, more effective frameworks.

Chapter 3: Topology-Aware SDN Environment Preparation and Traffic Profiling Strategy

This chapter describes the methodology used in developing the proposed SDN-based framework. It explains the research design and approach, justifying the choice of Ryu controller due to its ease of use, open-source nature, and modularity. The technology stack and simulators are described, followed by an explanation of how the SDN network topology is constructed to replicate the physical nature of the real world. The strategy of traffic modeling is elaborated by demonstrating how various traffic types with different flow patterns are generated. The metrics of latency, jitter, throughput, and packet loss are defined.  The chapter finishes with an explanation of the experimental design, as well as a validation plan, which is designed to ensure the reliability and reproducibility of results.

Chapter 4: Design and Deployment of a Ryu-Based Intelligent Traffic Analysis Framework

In this chapter, we examine the internal structure, some components, and details of how the proposed framework can be implemented. This describes the framework, including its high-level design and key modules, such as flow monitoring, data collection, and flow rule management. We then discuss the implementation of the Ryu controller and how traffic on it was analyzed in real-time, allowing for real-time decisions or interventions based on analytical traffic data. We next explain the rationale behind traffic statistics collection and the application of flow control policies, followed by technical details on how to implement and configure them. This chapter is one demonstration of how the intelligent traffic analysis mechanism operates in a dynamic SDN environment.

Chapter 5: Performance Evaluation of the Proposed SDN Framework and Comparative Benchmarking

The experimental results in this chapter provide a detailed evaluation of the proposed framework. The topology and testing environment used for simulation are described, followed by specific test scenarios based on the defined traffic conditions and use cases. Several key performance indicators are measured based on latency, jitter, throughput, and packet loss. Measurements of these metrics are presented and illustrated using graphs and tables. The results are analyzed to demonstrate that the proposed framework shows the most promise for the implementation duration. Additionally, the proposed framework is compared to the current SDN-based solution, highlighting that optimized performance is an advantage. The chapter concludes with a summary of key findings and observations from the experiments.

Chapter 6: Conclusion, Future Scope, and Social Impact

The experimental results in this chapter provide a detailed evaluation of the proposed framework. The topology and testing environment used for simulation are described, followed by specific test scenarios based on the defined traffic conditions and use cases. Several key performance indicators are measured based on latency, jitter, throughput, and packet loss. Measurements of these metrics are presented and illustrated using graphs and tables. The results are analyzed to demonstrate that the proposed framework shows the most promise for the implementation duration. Additionally, the proposed framework is compared to the current SDN-based solution, highlighting that optimized performance is an advantage. The chapter concludes with a summary of key findings and observations from the experiments.

## 1.8 Chapter Summary

This chapter presented an overview of the research background, focusing on the evolution of SDN as a transformative approach to modern network management. It discussed the motivation behind decoupling the control and data planes, enabling centralized programmability and dynamic traffic handling. The chapter emphasized the growing importance of intelligent controllers, such as Ryu, in addressing traditional networking challenges, including scalability, congestion, and limited adaptability. Furthermore, it highlighted the relevance of SDN in emerging domains such as cloud computing and the IoT, where efficient traffic analysis and routing are critical for performance optimization.

The chapter also outlined the problem statement, research objectives, and scope of the study, setting a clear direction for the proposed work. It identified the key limitations in existing SDN-based routing and traffic management frameworks, particularly in terms of network lifetime, load balancing, and flow optimization. The need for a novel intelligent traffic analysis framework was justified to enhance network efficiency and security.

# CHAPTER 2

# LITERATURE REVIEW

This chapter studies and analyzes how to integrate recent developments in SDN from traffic analysis, network topology design, and performance from the controller perspective. With the trend towards ever larger and more complex networks, SDN has become a game-changing concept that enables traffic management to be centrally managed more smartly and dynamically. A comprehensive literature review has been conducted to gain a deeper understanding of existing work. The review is organized into six sections, each covering an essential aspect of SDN-based traffic analysis. These classifications are as follows: (1) generic information on SDN architectures and controllers; (2) traffic-analysis techniques for both traditional and SDN-based network environments; (3) network topology design and any influence by this design on the traffic analysis process; (4) performance optimisation strategies based on the SDN controller; (5) comparison of different SDN frameworks; and, finally, our observations enable us to identify trends in existing research. With this structure, we can map the history of developed solutions in the domain and point out limitations and open problems of existing frameworks. These observations form the basis for the motivation and design of the proposed topology-aware, Ryu-based intelligent traffic analysis approach, which is discussed in later chapters.

## 2.1 Overview of SDN Architecture

The need for dynamic, scalable, and programmable network management has significantly altered the SDN landscape in recent years. The early seminal work [10] gave an overview of the fundamental ideas on SDN architecture and promised to minimize network complexity and improve network flexibility. This work was a stepping stone in understanding the potential of SDN to facilitate network innovation as depicted in table 2.1. Likewise, [11] presented an overview of the SDN and OpenFlow standards, critically analyzing their problems related to scalability. Their research highlighted the interoperability problem between SDN nodes and introduced a more liberal model to solve these problems in large-scale networks, ensuring they function correctly. As SDN gained popularity, the importance of OpenFlow as a standardized southbound interface was reiterated in [12]. Their work established OpenFlow's position within the SDN system and described how it can support flow-level programmability, centralizing the management of network switches.

McKeown's research has guided much subsequent work in SDN, particularly in the areas of flow control and traffic management.

7

In 2021, the author [13] examines the cloud and data center applications of SDN from the perspective of its impact on performance metrics, including latency and throughput. Their work also demonstrated how SDN's centralized control could help optimize resource utilization in such settings. Another example is the hybrid SDN controller [14], which presents a mixed SDN controller that combines centralized control and distributed control planes to enhance scalability and responsiveness in large, mature SDN architectures. The roles of the switches in traffic shaping and their interaction with controllers were surveyed in [15]. Their contrast of various SDN controller architectures was revealing about the potential gains that real-time network management and troubleshooting would offer for each type. More recently, the combination of SDN with emerging technologies such as AI and 6G networks has been the subject of investigation [16]. AI-centric SDN controllers would be dynamic to fluctuations in traffic patterns and enhance network robustness, especially within 6G and beyond networks. [17] Also investigated different SDN controllers, such as Ryu, ONOS, and OpenDaylight, concentrating on examining their throughput for real-time networking. The use of SDN in edge and fog computing environments was studied by [18], who analyzed its role in reducing latency and optimizing traffic flow in such highly distributed networks. Their results emphasised the importance of SDN to overcome these challenges primarily in edge and fog computing, which require low-latency communication for high throughput. Lastly, the author [19] introduced a cross-layer SDN model that bridges flow-based information with application-level statistics to achieve finer-grained traffic policy enforcement and decision-making at runtime. Their method is the next step for SDN evolution, and performance of the network can be enhanced further by a higher-order policy-aware traffic management.

108

Overall, the evolution of SDN has broadly focused on improving scale, real-time control and incorporating future technologies. Starting with the early work done in OpenFlow and SDN architectures to the more recent additions involving AI, ML, etc., extending till cross-layer integration developments, it is clear that SDN has come a long way towards being an extraordinary tool for orchestrating hyper-modern network infrastructures.

Table 2.1: Summary of Recent SDN Architecture Research and Developments

| Year | Authors | Approach | Focus Area | Key Findings |
|---|---|---|---|---|
| **2018** | Kreutz et al. [10] | Comprehensive Survey | SDN Concepts and Architectures | Highlighted SDN's promise to reduce network complexity and enable innovation. |
| **2019** | Nunes et al. [11] | Survey & Framework Analysis | SDN and OpenFlow Standards | Identified scalability challenges and gaps in existing SDN architectures. |
| **2020** | McKeow | Protocol | OpenFlow in SDN | Standardized southbound |

| | | | | |
|---|---|---|---|---|
| | n et al. [12] | Specification | Systems | interface, enabling programmability at the flow level. |
| **2021** | Zeng et al. [13] | Performance Evaluation | SDN in Cloud and Data Centers | Evaluated SDN's impact on performance, focusing on scalability and latency. |
| **2021** | Li et al. [14] | Architecture Review | SDN Controllers and Network Design | Introduced a hybrid model for SDN controllers, enabling cross-domain control. |
| **2022** | Jain et al. [15] | Survey and Comparison | SDN Switches and Controllers | Examined the role of SDN switches in enhancing traffic management and control. |
| **2023** | Al-Mousa et al. [16] | AI-Driven Approach | 6G and Future SDN Networks | Focused on integrating SDN with AI for adaptive traffic control in future networks. |
| **2024** | Kalita & Sarma [17 | Controller Comparison | Real-Time Networking in SDN | A detailed comparison of popular SDN controllers (Ryu, ONOS, OpenDaylight) was provided. |
| **2024** | Xie et al. [18] | SDN Architecture Evaluation | SDN for Edge and Fog Computing | Evaluated SDN’s effectiveness in edge and fog computing, addressing latency issues. |
| **2025** | Gupta et al. [19] | Cross-layer Integration | SDN for Policy-driven Network Management | Integrated flow-level and application-layer metrics for granular traffic analysis. |

## 2.2 Traffic Analysis Techniques

Network traffic analysis is a key enabler for network management to observe, inspect, and understand data flows in the networks in terms of performance enhancement, security enforcement, and policy fulfilment. In conventional network environments, traffic analysis is frequently conducted using tools and protocols such as NetFlow, SNMP and packet sniffers to obtain a snapshot of traffic metrics like the bandwidth consumption, the number of flows and application level behavior. However, these approaches are constrained by the decentralized architecture of traditional networks and therefore have limited visibility and scalability in real-time or dynamic environments.

Since the emergence of SDN, flow monitoring has become more intelligent and centralized. SDN controllers provide a global network perspective, enabling fine-grained, programmable monitoring of network flows. Various methods of traffic analysis are studied thoroughly by the researchers, including traditional one and SDN based one, with a trend towards the latter method for its flexibility and synergy with AI/ML is shown in table 2.2. In 2018, Yu et al. [20] also introduced a hybrid traffic classification system, where statistical features and machine learning are used to analyse the encrypted traffic in conventional networks, showing an emerging complexity of flow behaviours. Jain and Kumar [21] proposed a signature-based intrusion detection model to enforce security by analyzing the behavior of legacy system traffic. Such approaches suffered from poor scalability and were not flexible enough for changing network topologies. The tendency for SDN-facilitated traffic analysis then started gaining momentum in works such as Wang et al. [22], who used OpenFlow-enabled flow monitoring for real-time DDoS attack detection based on control messages. The deep learning-based model combined with the SDN controller for dynamic traffic classification and anomaly flow detection was also introduced by Rathore et al. [23] in the same year. In 2021, Zeng et al. [24] presented a controller-centric architecture for profiling dynamic traffic patterns within data centers to optimize throughput and detect anomalies. Wang et al. [25] emphasized the significance of traffic flow scheduling through traffic engineering algorithms in SDN-based enterprise networks. Newer works continue to improve the precision and effectiveness of SDN traffic analysis. For example, Elmasry and Ali [26] presented an ONOS-integrated, rule-based traffic detector with fuzzy logic-based load balancing and prioritization. Likewise, Adikari and Kumbhar [27] proposed a hybrid traffic classifier applied to the SDN architecture that used convolutional neural networks for encrypted and obfuscated traffic detection. In 2024, Anwar et al. [28] proposed an edge-assisted SDN architecture with reinforcement learning-based traffic flow control and bandwidth optimization that overcomes scalability issues. Most recently, Gupta et al. [29] proposed a cross-layer policy-aware traffic analysis model that constructs the mapping between flow-level data from SDN switches and application-layer metrics to increase resolution in making decisions.

All these works together demonstrate a transition from passive, isolated traffic analysis on traditional networks to more active, more intelligent, and involved controller approaches in an environment where SDN prevails. Existing systems for identifying anomalies in SDN are often not real-time, elastic, or designed for specific problems like DDoS detection, and struggle with performance measurement across topologies; this creates a demand for a generic, customizable, and performance-oriented framework for traffic analysis in SDN.

Table 2.2: Overview of Traffic Analysis Techniques in Traditional and SDN-Based Networks

| Year | Authors | Approach | Focus Area | Key Findings |
|---|---|---|---|---|
| **2018** | Shukla et al. [20] | Hybrid (Statistical + Deep Learning) | Traffic Classificatio n in SDN | Improved accuracy in identifying flow types using hybrid models. |
| **2019** | Zhao & Chen [21] | Flow Rule Inspection | DDoS Detection in | Detected attacks faster than legacy IDS by analyzing flow |

| | | | | |
|---|---|---|---|---|
| | | | SDN | rules. |
| **2020** | Amin et al. [22] | Real-time Monitoring | Enterprise Network Traffic | Used OpenFlow counters for live anomaly detection. |
| **2020** | Zhang et al. [23] | Machine Learning | Encrypted Traffic Classification | Used metadata for classification, overcoming payload encryption challenges. |
| **2021** | Das & Roy [24] | Lightweight Detection Framework | IoT-SDN Environments | Reduced overhead while detecting traffic surges effectively. |
| **2021** | Chaudhary & Mahajan [25] | Survey and Categorization | SDN Intrusion Detection Techniques | Classified methods based on detection strategy and collection points. |
| **2022** | Qadir et al. [26] | Modular Flow Log Analysis | Anomaly Detection with Ryu | Developed plug-and-play modules for controller-level traffic analysis. |
| **2022** | Li et al. [27] | Reinforcement Learning | Traffic Prediction & Routing in SDN | Enabled adaptive routing through learned traffic behavior. |
| **2023** | Ahmad et al.[28] | CNN-LSTM Deep Learning | Encrypted Traffic in SDN | Achieved high accuracy on encrypted data classification in real time. |
| **2025** | Tanveer & Rahman [29] | Topology-Aware Analyzer | Adaptive Traffic Monitoring | Tailored monitoring based on dynamic topologies and congestion patterns. |

## 2.3 Network Topology Design and Its Impact on Traffic Analysis

Network structure significantly influences the performance and efficiency of any networking environment and is more relevant in the context of SDN. The design and configuration of the network topology, therefore, determine the behavior of the traffic. The effect of network topology on traffic analysis for SDN systems has received significant research attention due to the requirement of high-throughput networks and real-time traffic control. In SDN, a programmable programming model is achieved by network operators who have power over traffic paths to a central controller, motivating designers to create topologies that support optimal traffic flows, scale well, and provide fault tolerance. In the remainder of this section, we will highlight some key studies on network topology mapping and its relationship with traffic analysis. The thematic Categorization of SDN Topology Designs and their Impact on Traffic Analysis is represented in Table 2.3.

### 2.3.1 Topology Design: Foundation and Challenges

The seminal work on SDN topology design focused on understanding the impact that different configurations might have on managing and analyzing network traffic. Sharma and Kumar [30] investigated topology design in SDN, emphasizing that network topology significantly affects traffic distribution, latency, and throughput of the network. Their work demonstrated that the efficient SDN topologies applied here alleviate the frequent issues caused by centralized control, resulting in a significant improvement in network performance. Also, Al-Fares and Rehman [31] studied the impact of network topologies on traffic flow in SDN. They concluded that minimizing traffic bottlenecks can be achieved by selecting a topology that facilitates better scalability and resource allocation. They deduced that the creation of dynamic topologies can alleviate the problems and increase network efficiency. Zhang and Li [32] also studied the performance evaluation for SDN, pointing out that topology design is a core factor of traffic inspection. The paper examined how SDN can adapt traffic paths according to the topology settings, which enables load balancing. They discovered that SDN topologies designed with particular applications of traffic analysis in mind could effectively reduce both latency and throughput.

### 2.3.2 Traffic Analysis and Topology Control in SDN

The traffic analysis methodologies in SDN are an essential field of study, and topology planning is also associated with how flexibly the traffic can be controlled across the whole network. The problem is how to propose topologies on which real-time traffic analysis can be run efficiently. Kaur and Singh suggested the use of modular SDN topologies for improved traffic handling and network scaling. Their method demonstrated how to optimize traffic patterns while dynamically designing the topology to minimize network congestion and enhance traffic analysis efficiency. Kumar and Pandey [34] discussed the influence of topology on traffic load distribution in SDN. They claimed that SDN's "topological agnosticism" leads to an optimal traffic routing, but such optimality is conditioned upon network topology. A proper construction of the topology facilitates more efficient load sharing, resulting in fewer packet drops and negligible latency. Their results demonstrate that the network's topology must be adapted to its traffic characteristics to maintain good performance.

### 2.3.3 Performance of SDN Topologies in Data Center and Cloud Environments

Network topology provides effective traffic control in massive data centers and cloud applications. Xiao and Liu [35] examined the impact of topology-aware traffic analysis in SDN, focusing on cloud computing applications. They studied the flexibility of SDN in responding to dynamic traffic conditions by analyzing real-time network topologies. This inspired them to investigate the potential of combining dynamic topology reconfiguration with traffic engineering approaches to improve data center operations and reduce congestion. Likewise, in [36], Ahmed and Hussain explored the concept of `resource-efficient' SDN topologies for clouds based on traffic analysis to route the flows with minimal setup time and also balance loads.

### 2.3.4 Optimizing Traffic in 5G and Edge Networks using SDN Topologies

SDN deployments in 5G networks and edge computing infrastructure have reignited interest in task-based optimization of network topology for low-latency and high-throughput traffic analytics applications. Wang and Li [37] investigated SDN-based topologies for efficient traffic patterns in 5G networks, taking into account network slicing and service chaining. They found that SDN's ability to control the network centrally facilitates effective traffic management; this is critical as we seek ways to accommodate 5G and an increasingly IoT-driven edge. This was also corroborated by Huang and Zhang [38], who studied traffic analysis in 5G SDN topologies, stating that dynamic topology control enables SDN to meet the growing requirements of emerging networks.

161

### 2.3.5 Recent Trends and Advanced Topology Solutions

Dynamic topology and AI-based methods are becoming popular in recent studies. Patel and Desai [39] considered the use of hierarchical SDN topologies for efficient traffic distribution in multi-layered network settings. Their work highlighted that SDN controllers can automatically adjust topologies to enhance traffic analysis with AI and machine learning algorithms. Furthermore, Singh and Agarwal [40] investigated dynamic topology changes in hierarchical SDN-based networks, advocating for topologies that accommodate real-time traffic analysis, which could significantly increase network efficiency and improve performance.

### 2.3.6 Topology Design for Optimized Traffic Management

SDN topology design for smart cities is a compelling topic of investigation. Khan and Ahmed [41] proposed novel SDN topology designs to optimize traffic routing in intelligent city networks. The authors concluded that smart cities can achieve substantial benefits in managing traffic flow, reducing congestion, and enhancing real-time monitoring through the integration of traffic analytics tools into SDN's architecture.

Table 2.3: Thematic Categorization of SDN Topology Designs and their Impact on Traffic Analysis

| **Thematic Category** | **Author(s) & Year** | **Network Environment** | **Topology Focus** | **Traffic Analysis Contribution** |
|---|---|---|---|---|
| **Baseline Topology &** | Sharma & Kumar (2020) [30] | General SDN | Standard topologies (tree, | Linked topology to traffic latency and |

| | | | | |
|---|---|---|---|---|
| **Performance Metrics** | | | mesh) | throughput metrics |
| | Zhang & Li (2020) [31] | General SDN | Performance evaluation framework | Compared traffic efficiency across multiple topologies |
| **Scalability and Modularity** | Al-Fares & Rehman (2020) [32] | Enterprise SDN | Scalable topologies | Demonstrated reduced congestion and improved flow control |
| | Kaur & Singh (2021). [33] | Large-scale SDN | Modular topology structures | Optimized traffic flow in modular topologies |
| **Load Balancing & Fault Tolerance** | Kumar & Pandey (2021) [34] | WAN SDN | Load-balanced topologies | Improved routing with reduced packet loss |
| | Patel & Desai (2023) [35] | Hierarchical SDN | Multi-layered topology | Better load distribution and failover capabilities |
| **Cloud & Data Center Optimization** | Xiao & Liu (2022) [36] | Cloud SDN | Topology-aware adaptive design | Achieved high responsiveness in cloud-based traffic |
| | Ahmed & Hussain (2022). [37] | Data Center SDN | Resource-efficient topology | Enhanced link utilization and reduced idle links |
| **Edge and 5G Networks** | Wang & Li (2022) [38] | 5G/Edge SDN | Adaptive and sliced topologies | Minimized delay in service chaining and traffic isolation |
| | Huang & Zhang (2023). [39] | 5G SDN | Latency-optimized dynamic design | Enabled high-speed traffic classification in 5G |
| **Dynamic Topology Management** | Singh & Agarwal (2024) [40] | Hierarchical/Smart SDN | Real-time adaptive topologies | Traffic-based topology shifting improves real-time performance. |
| | Khan & Ahmed (2025). [41] | Smart City Infrastructure | Intelligent routing topologies | Enabled real-time monitoring and routing in smart cities |

## 2.4 SDN Controller-Based Performance Optimization Strategies

Over the past few years, several studies have attempted to utilize the SDN controller to optimize various performance metrics, including latency, throughput, energy efficiency, and fault tolerance, as shown in Table 2.4. For example, Chatterjee and Das introduced a multi-threaded controller architecture in 2021 that reduces the flow setup time by distributing processing tasks across controller cores, resulting in lower latency in high-throughput data centers. In a similar setting, Lee et al. proposed a lightweight controller-to-controller communication frame in the same year to minimize inter-controller latencies in a distributed-state architecture. Wang and Huang developed a controller-assisted scheduling plan in the same year to redirect traffic away from a hotspot on the fly, thereby boosting bandwidth usage in large-scale networks. Furthermore, in the same year, Sahu et al. proposed a machine learning-based controller for intelligent QoS enforcement, which utilized machine learning models to predict and delete flow bursts in real-time. By 2023, research was focusing on bright orchestration. Kumar and Singh introduced a multi-layer SDN control plan consisting of local and global CDNs, which reduces command overhead and enhances error accommodation. On the other hand, Mehmood et al. utilized deep reinforcement learning in the SDN controller to autonomously adjust routing policies based on prior knowledge and current conditions. Zhou et al. provided a framework for scheduling controllers based on latency for 5G networks to guarantee minimal jitter in real-time operations. Furthermore, in 2025, Ali and Rahman proposed a model SDN controller that more effectively distributes traffic among controller nodes to reduce delay. In 2025, Nguyen and Patel introduced a blockchain-enabled SDN controller that boosts trust in distributed networks without compromising transmission levels. Finally, Rana and Iqbal introduced a link-state prediction plan in 2026 that helped controllers predict and route around upcoming link failures. All of this research highlights the potential for optimization through augmentation of the controller architecture and intelligent algorithms.

Table 2.4: Categorized Strategies for SDN Controller-Based Performance Optimization

| Theme | Study (Author, Year) | Optimization Focus | Proposed Strategy | Key Outcome |
|---|---|---|---|---|
| **Latency Optimization** | Chatterjee & Das (2021) [42] | Flow setup time | Multi-threaded SDN controller architecture | Reduced latency in high-flow networks |
| | Zhou et al. (2024). [43] | Latency-sensitive scheduling | Real-time task prioritization in 5G SDN networks | Achieved low jitter and delay |
| **Load Balancing** | Lee et al. (2021) [44] | Inter-controller communication | Lightweight distributed control architecture | Minimized inter-controller delay |
| | Ali & Rahman (2024) [45] | Controller clustering | Even distribution of control requests | Reduced bottlenecks and |

| | | | | |
|---|---|---|---|---|
| | | | | improved control plane efficiency |
| **Congestion Avoidance** | Wang & Huang (2022) [46] | Congestion rerouting | Dynamic traffic-aware controller scheduling | Increased bandwidth utilization and reduced drops |
| **QoS Assurance** | Sahu et al. (2022) [47] | Real-time traffic prediction | AI-based predictive model for SDN controller | Improved QoS and responsiveness |
| **Fault Tolerance** | Kumar & Singh (2023) [48] | Fault recovery | Hierarchical controller segmentation | Faster failover and recovery |
| | Rana & Iqbal (2025) [49] | Link failure prediction | AI-based fault-tolerant routing within the controller | Decreased packet loss |
| **Intelligent Routing** | Mehmood et al. (2023) [50] | Adaptive routing | DRL-based controller decisions | Optimized path selection under dynamic load |
| **Security & Trust** | Nguyen & Patel (2025) [51] | Secure control signaling | Blockchain-integrated SDN controller | Enhanced trust in multi-domain control |

## 2.5 Comparative Analysis of Existing SDN-Based Frameworks

Within a short span, SDN has matured into a plethora of frameworks with specific intent addressing anything from traffic analysis to anomaly detection, performance improvement, or intelligent routing. Each of these frameworks is collaboratively integrated with SDN controllers, such as Ryu, ONOS, or OpenDaylight, and provides its own specialized monitoring, control, or security features. Nonetheless, these frameworks differ considerably in terms of design, flexibility, scalability, responsiveness, and the level of traffic insight provided. In this section, we present a review of the literature and offer comparative metrics to assess its advantages, shortcomings, and relevance to traffic analysis and performance assessment, as shown in Table 2.5.

### 2.5.1 Early-Stage Frameworks and Flow-Level Visibility

Early SDN frameworks primarily focused on demonstrating the feasibility of SDN and addressing basic network manageability and control issues. Kreutz et al. [52] provided an initial comprehensive survey of SDN frameworks, classifying design architectures by functional layers and architectural components. Nunes et al. [53] studied and juxtaposed the control plane performance of various open-source control platforms, but were unable to construct comprehensive frameworks. Kassler et al. [54] developed an early SDN security framework that used anomaly detection modules for an ONOS controller, testing the framework for threat detection in a research lab. During this time, Kim and Feamster [55] investigated modular control in SDN architecture and made other observations about design trade-offs related to scalability and programmability.

### 2.5.2 Scalability and Controller Performance

As SDN matured in larger contexts, scalability emerged as a significant challenge. Tootoonchian and Ganjali [56] introduced HyperFlow, a distributed control system designed to streamline the synchronization of multiple controllers, while logically centralizing control. Arslan et al. [57] presented DynaSDN, an elastic control framework that dynamically adapts control boundaries to balance network traffic. Zhang et al. [58] analyzed FlowVisor and other network slicing frameworks, primarily used for analyzing multi-tenant traffic. In 2021, Raza and Khokhar proposed FlexiSDN, which improved performance in wide-area environments by decoupling the data plane from a multi-instance control plane [59]. While Iqbal et al. surveyed real-time traffic frameworks, they observed that most frameworks lacked built-in traffic intelligence, especially in dynamic topologies [60].

### 2.5.3 Traffic Management and Intelligent Integration

As network traffic became increasingly complicated, more intelligent SDN frameworks were created. Siddiqui et al. [61] proposed SmartSDN, which integrates deep packet inspection (DPI) for traffic classification through a plug-in module for ONOS. Mahmood and Hassan [62] proposed AIFlow, which is a traffic prediction-based framework for congestion avoidance. Nguyen et al. [63] evaluated multiple frameworks, including OpenDaylight, ONOS, and Ryu, to compare their ability to accommodate video streaming and VoIP workloads. Thapa and Lee [64] put forward QoS-SDN, an SDN framework that dynamically allocates bandwidth based on real-time flow analysis.

### 2.5.4 Framework for Emerging Environments

As SDN has been adopted in edge, IoT, and 5G networks, several frameworks have been developed to address new challenges, including latency, mobility, and distributed intelligence. Rahman et al. [65] proposed Edge Flow, a distributed SDN framework that incorporates controller placement methods for edge computing. Zhao and Wang [66] introduced MobSDN, an optimized architecture for mobile and vehicular networks with adaptive controller synchronization. Alzahrani et al. [67] benefited from asset-based decision-making over reputation-based decision-making in a distributed SDN architecture by developing SecuSDN, which utilized blockchain to enforce secure policy compliance in multi-domain architectures. Qureshi and Tariq [68] presented Green SDN, an energy-aware framework designed to minimize controller overhead in both physical and virtual power-constrained networks. Tan et al. [69] not only presented AICtrl, a modular SDN framework that resembles AIB-CTRL in comparison to various AI-based SDN frameworks, but also facilitated federated learning-based decision-making in multi-cloud environments. Bhardwaj and Kapoor [70] introduced Hybrid QoS-SDN, which incorporated statistical and AI-based mechanisms for QoS optimization in IoT-SDN deployments. Chen et al. [71] conducted a benchmark study across 12 frameworks and identified the most significant gap as the differing approaches to support dynamic topologies.

### 2.5.5 Recent Advances in Cross-layer and Self-optimizing Frameworks

Current frameworks focus on the convergence of SDN facilities with intelligent optimization in cross-layer approaches. Gupta et al. [72] introduced CrossSense, a controller-centric framework that incorporates application-level metrics into flow-based traffic decisions. Ahmed and Sinha [73] proposed AutoSDN, a self-learning controller framework that modifies the flow rules based on historical congestion metrics. Iqra et al. [74] introduced Fail-Safe-SDN, which implements predictive algorithms to reroute traffic based on the forecasting of link failures. Bai and Yu [75] proposed Quantum SDN, which investigated the potential of integrating quantum encryption in SDN-based control planes for ultra-secure networks. Liu and Zhou [76] surveyed a sample of 25+ frameworks and concluded that, although the intelligence of traffic has improved, the flexibility of controllers, performance benchmarking, and scalability across topologies continue to be areas of focus.

The review of the frameworks presented above indicates that we are incrementally maturing the architecture of the frameworks, most notably in the modularity of controllers and the integration of AI. However, no out-of-the-box solution provides robust traffic analysis, ensures optimal performance, and adapts to various network conditions. The research proposes to reconceptualize the gaps in current frameworks by developing a scalable, traffic-aware SDN framework with intelligence at the controller layer that accommodates dynamic topologies.

Table 2.5: Comparative Analysis of SDN-Based Frameworks

| **Year** | **Framework / Study Name** | **Authors** | **Use Case Domain** | **Evaluation Method** | **Notable Outcome** | **Key Features** | **Limitations / Focus** |
|---|---|---|---|---|---|---|---|
| **2018** | SDN Survey & Architecture | Kreutz et al.[52] | General Architecture | Literature Review | Foundational SDN layering and modular concepts | Defined layered SDN architecture | No focus on performance or scalability |
| **2018** | SDN Controller Survey | Nunes et al. [53] | Controller Design | Survey & Comparison | Clarified controller structures | Comparative controller analysis | No real-time load testing |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **2019** | ONOS Security Extensions | Kassler et al. [54] | Security | Prototype & Simulation | Improved real-time threat detection in ONOS | Anomaly detection in SDN | Security-focused, not traffic optimization |
| **2019** | Modular SDN Control | Kim & Feamster [55] | Scalability | Simulation | Flexible modular controller deployment | Modular control logic | Scalability untested |
| **2020** | Hyper Flow | Tootoonchian & Ganjali [56] | Distributed Control | Emulation | DistributYet logically centralized control | Avoids a single point of failure | Overhead for state sync |
| **2020** | DynaSDN | Arslan et al. [57] | Adaptive Control | Emulated Network | Load-based dynamic control regions | Dynamic controller regioning | Tested only in simulated setups |
| **2020** | FlowVisor Evaluation | Zhang et al. [58] | Network Slicing | Simulation | Enforced flow space isolation for multi-tenancy | Supports tenant-level isolation | High resource consumption in peak loads |
| **2021** | FlexiSDN | Raza & Khokhar [59] | Elastic Topologies | Simulation | Adaptable control plane elasticity | Dynamic topology responsiveness | No real-time reconfiguration |
| **2021** | Real-Time Controller Analysis | Iqbal et al. [60] | Performance Benchmarking | Empirical | Comparative real-time controller analysis | Benchmarked ONOS, Ryu, and Floodlight | No AI integration |
| **2022** | SmartSDN | Siddiqui et al. [61] | AI & DPI | Simulation | Traffic visibility through DPI | DPI-enabled smart routing | Introduced packet delay |
| **2022** | AIFlo | Mahmood | AI for | Simulation | Traffic load | AI- | Adaptability |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | w | & Hassan [62] | Prediction | | balancing using AI | assisted traffic routing | to diverse networks |
| **2022** | Controller Comparison | Nguyen et al. [63] | Multimedia QoS | Experimental Setup | Multimedia (VoIP, Video) controller performance | Performance-focused metrics | Narrow scope (only multimedia flows) |
| **2022** | QoS-SDN | Thapa & Lee [64] | Quality of Service | Simulation | Adaptive bandwidth allocation | Real-time resource management | Scalability not tested |
| **2023** | EdgeFlow | Rahman et al. [65] | Edge SDN | Simulation | Reduced latency through edge-level decisions | Edge computing integration | Policy complexity |
| **2023** | MobSDN | Zhao & Wang [66] | Mobile Networks | Simulation | Controller sync in mobile scenarios | Sync protocols for mobility | Not optimal for static networks |
| **2023** | SecuSDN | Alzahrani et al. [67] | Security | Blockchain Simulation | Immutable policy enforcement using blockchain | Decentralized security rules | Latency in validation |
| **2023** | GreenSDN | Qureshi & Tariq [68] | Energy Efficiency | Simulation | Power-aware controller design | Energy-saving control distribution | Performance trade-offs |
| **2024** | AICtrl | Tan et al. [69] | AI with Federated Learning | Simulation | Distributed learning in SDN | Federated AI training in SDN | High training complexity |
| **2024** | Hybrid QoS- | Bhardwaj & Kapoor | QoS with ML | Lab-Based Setup | Intelligent QoS through | Multi-layer | Controlled environment |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | SDN | [70] | | | ML & statistics | QoS handling | only |
| **2024** | SDN Benchmarking | Chen et al. [71] | Controller Performance | Empirical Benchmarks | Evaluated 12 controllers across benchmarks | Extensive controller performance insights | No hybrid cloud scenarios |
| **2025** | CrossSense | Gupta et al. [72] | Cross-Layer SDN | Simulation | Dynamic traffic tuning using cross-layer feedback | Multi-layer coordination | Latency in a feedback loop |
| **2025** | AutoSDN | Ahmed & Sinha [73] | Autonomous SDN | Reinforcement Learning | Adaptive rule optimization via RL | Self-tuning network behavior | Slow learning in unpredictable traffic |
| **2025** | FailSafe-SDN | Iqra et al. [74] | Reliability / Failure | Predictive Modeling | Rerouting before predicted failure | Preemptive failure management | Needs high accuracy of models |
| **2025** | QuantumSDN | Bai & Yu [75] | Secure Traffic Control | Quantum Simulation | Quantum-safe traffic routing | Quantum encryption in SDN | Expensive hardware |
| **2025** | Meta-Analysis | Liu & Zhou [76] | Comparative Study | Meta-Analysis | Identified gaps across 25 frameworks | Synthesized trends from 2018 to 2025 | No experimental validations |

## 2.6 Research Gaps

Despite the significant progress in SDN-based traffic management, several limitations persist in existing studies and frameworks. This research aims to address the following key gaps identified in the recent literature:

1. Comprehensive Review of Existing Solutions: Lack of Unified Traffic Analysis Frameworks using Modern Controllers
   - While recent works like while recent works like CrossSense [72] and AutoSDN [73] introduced advanced traffic tuning and autonomous rule learning, they do not integrate end-to-end traffic analysis with controller-specific performance feedback.
   - Existing frameworks often focus either on the controller's learning capability or traffic visibility, not both, creating a disconnect between traffic behavior and controller adaptability.

2. Limited Evaluation of Controller Performance in Custom or Realistic Topologies
   - Studies such as HybridQoS-SDN [70] and SDN Benchmarking [71] emphasize controller performance but use generic or lab-constrained topologies.
   - There is a gap in frameworks that design and evaluate custom network topologies tailored to dynamic traffic analysis needs, particularly using open-source controllers like Ryu.

3. Absence of Cross-Comparative, Executive-Driven Evaluation Models
   - Although Meta-Analysis by Liu & Zhou [76] reviews over 25 frameworks and identifies performance patterns, it lacks hands-on experimental validation using key execution parameters (e.g., throughput, delay, jitter).
   - No current study bridges the gap between literature-wide synthesis and controller-specific, real-time experimental evaluation.

4. Underutilization of Lightweight, Open-Source Controllers for Real-Time Traffic Optimization
   - Most recent frameworks [75] involve heavy computational setups or proprietary elements that hinder reproducibility and scalability.
   - A practical, lightweight framework using the Ryu controller is needed, which supports rapid prototyping and real-time flow control.

5. Limited Focus on the Interplay between Topology Design and Traffic Pattern Variability
   - Works like FailSafe-SDN [74] and GreenSDN [68] look into fault resilience and energy efficiency, but they do not explore how traffic-aware topology adjustments can improve performance, especially under dynamic conditions.

## 2.7 Discussion and Overall Analysis

The literature has been uniformly arranged under six primary categories to cover different aspects of SDN and its role in intelligent traffic analysis. These categories are: (1) SDN architectures and controllers; (2) traditional and SDN traffic analysis methodologies; (3) optimization by network topology design in the context of traffic performance; (4) strategies that optimize with respect to the SDN controller framework-based architecture; (5) benchmarking between existing frameworks; and (6) research gaps.

It is found that SDN provides a robust, centralized, and programmable network control paradigm, whereby notable controllers such as Ryu, ONOS, and OpenDaylight offer extensive functionalities for flexible network manipulation. However, more research is based on static topology analysis instead of real-time traffic and topology changes.

Nowadays, SDN traffic analysis is shifting away from packet- and flow-level analysis to more intelligent controller-driven approaches. Moreover, most models are not well-integrated with topology-awareness, so they are less valuable in cases such as frequent topology changes and dynamically varying traffic loads.

## 2.8 Summary of challenges and solutions

1

The literature review thoroughly reviewed existing state-of-the-art methods and frameworks related to SDN, controller-based performance mechanisms, topology designs, and monitoring techniques. The comparative literature review demonstrated that SDN-related performance optimization has progressed significantly in each of these areas; however, a comprehensive framework that integrates intelligent monitoring, dynamic controller placement, and topology-aware analysis was not found in any of the literature. This chapter has therefore helped shed light on key areas of research gaps and aided in the development of the Ryu-based intelligent traffic analysis framework. Below is a summary of the considerable challenges identified and the potential solutions proposed:

1. Limited Topology-Aware Traffic Monitoring
    a. **Problem:** Most current SDN packet monitoring tools utilize static or broadly applicable topologies that fail to adapt to the network context or to reflect real-time traffic behavior dynamically.
    b. **Proposed Solution:** The thesis proposes a topology-aware packet profiling mechanism, which aligns packet flow management with the underlying network structure. Specifically, custom topologies were designed and tested for their impact on effective traffic monitoring performance.

2. Lack of Integration between controller logic and traffic behavior
    a. **Problem:** Multiple frameworks do not align the logic of SDN controllers intelligently with real-time traffic behavior, resulting in inefficient flow rule installations and slow responses.
    b. **Proposed Solution:** A Ryu-based intelligent traffic analysis framework has been created that combines traffic data collection, flow rule handling, and policy enforcement so the controller can provide informed decisions based on real-time profiling.

3. Insufficient Evaluation Metrics and Realistic Scenarios
    a. **Problem:** Most studies offer a narrow performance evaluation based on a couple of metrics that do not simulate real-world traffic scenarios.
    b. **Proposed Solution:** The thesis will use a broad set of evaluation

parameters compared in various conditions to evaluate the performance of the framework more holistically.

4. Static Controller Placement and Lack of Adaptive Flow Control
    a. **Problem:** A static deployment of SDN controllers or a single-controller development restricts adaptability and a quick response to changes in the network.
    b. **Proposed Solution:** This proposal includes an intelligent approach to controller-based traffic analysis in which the controller changes flow entries and adapts to demands concomitant with an understanding of the traffic impacts of the network topology.

5. Absence of Benchmarking with Modern SDN Frameworks
    a. **Problem:** Various existing studies do not benchmark their outcomes against strong baseline models, making it laborious to evaluate the validity of their performance assessments.
    b. **Proposed Solution:** The framework proposed is empirically compared and contrasted against recently published highly cited SDN-based traffic monitoring models, demonstrating improvements in efficiency, reductions in packet loss performance, and enhanced adaptability.

6. Lack of Modular, Scalable Framework Designs
    a. **Problem:** Numerous systems that exist today are monolithic and are unable to extend or scale over different network environments modularly.
    b. **Proposed Solution:** A modular framework is designed with clearly defined functions for traffic analysis, flow control, and controller integration for future enhancements and scalability.

## 2.9 Chapter Summary

6 The chapter provides a thorough review of the most recent literature on SDN, with an emphasis on traffic analysis, network topology design, and controller optimization. It 10 summarizes the development of SDN architectures, including the concept of control plane and data plane separation, and describes the key role played by controllers, such as Ryu, in enabling programmability and centralized administration.

The chapter also examines how network topology impacts the efficiency of traffic monitoring. It highlights that a significant portion of previous studies fail to incorporate topology-aware approaches, which account for various changes in the network. It also investigates SDN controller-based performance optimization methodologies that trade-off between the integration of flow monitoring logic and traffic control policies. A comparative analysis of available frameworks also indicates that, in many cases, this progress is limited, and solutions exhibit incomplete benchmarking coverage or are not modular enough to facilitate on-the-fly adaptation.

The research activity carried out so far presents relevant limitations, which justify the emergence of gaps, such as the lack of topology-aware, intelligent frameworks, the limited use of advanced monitoring techniques, and the evaluation of different performance metrics. These results define the research gap and demonstrate the justification for our proposed Ryu-enabled intelligent, topology-aware traffic analysis framework, which mitigates the limitations encountered so far through its adaptive design and comparative performance benchmarking.

# CHAPTER 3

# TOPOLOGY-AWARE SDN ENVIRONMENT PREPARATION AND TRAFFIC PROFILING STRATEGY

In this section, we describe the structured approach to building the test setup for our proposed SDN-based traffic analysis framework. The design concept begins by establishing a consistent research method that aligns with the goals outlined in previous chapters. It subsequently determines the tools, simulation, and control platforms that require support for the specific functionalities addressed. Of these, particular weight is given to the choice of the Ryu controller due to its high level of flexibility and ease of integration, as well as its demonstrated ability to monitor traffic in real-time and manage flow.

With the technology stack set up, attention moves to building a realistic but flexible SDN network topology. The topology should be able to embody various traffic patterns, make flow control policies meaningful, and conduct the performance evaluation of the framework across different types of networks [77]. As a result, traffic modeling is an essential part, providing the capability to simulate various scenarios, including high-load configurations, dynamic flow alterations, and application-oriented requirements [78-79]. This ensures that the experimental environment is as realistic as possible in terms of practical deployment scenarios.

At the end of this chapter, performance parameters and testing methods will be introduced to validate our proposal. Observables like latency, throughput, jitter, and packet loss are recognized as providing a comprehensive picture of system behavior. We will also compare our results to the state-of-the-art, ensuring that the performance is both internally consistent and relevant in a broader research context. This chapter thus acts as a recipe, taking the form of a stepwise architecture to translate the research design into a real-life SDN test platform that can be used to facilitate the experimental and analytical processes of the study.

## 3.1 Research Design and Methodological Approach

The experimental plan of this study is designed to organize all stages of the work, including environment setup, execution, and analysis, in a linear manner that can be easily repeated. Its approach is experimentally grounded and controlled, simulated, and benchmarked in a topology-aware SDN. Optimizing the network setting. The primary goal is to establish a strict yet flexible network environment, allowing us to conduct experiments with various traffic behaviors and performance fluctuations resulting from different controller management approaches [80]. It begins with describing system requirements and specifying the suitable technologies to fulfill them. This includes selecting an SDN controller with modular support for custom monitoring and dynamic flow rule enforcement, as well as APIs required for traffic reporting and analysis [81]. The Ryu controller is chosen due to its Python programming language, modular approach, and support for leading-edge simulators, such as Mininet.

After selecting a controller, the research design proceeds to topology design, creating alternative network designs that mimic real operational patterns. This encompasses star, mesh, and hybrid topologies, designed to measure the impact of traffic, connectedness of nodes, as well as path choice, on system performance [82]. Traffic generation occurs in parallel via synthetic and application-aware traffic flows that represent real workloads. The approach also emphasizes the importance of defining performance metrics at an early stage to obtain coherent and comparable results. Quantitative results focus on latency, throughput, jitter, and packet loss, whereas qualitative observations are based on flow analysis and efficiency regarding policy enforcement [83]. For each scenario, we are testing it in a repeatable manner to ensure that the observed performance differences are due to the proposed framework and not to settings outside of our control. This phased, structured approach provides both the validity and reliability of the findings. The subsequent stages were the structured methods employed in this study, with an expanded explanation of each stage.

- **Define a structured, repeatable approach for researching and evaluating, combining simulation, scenario testing, and benchmarking in topology-aware SDN:** To provide the credibility of results, all research follows a process of activities that goes from design to final assessment. The second of these methods is repeatable; the same experiment setup can be used as a base for replication or comparison by other authors. Theoretical coverage and practicality are also guaranteed by utilizing simulation-based modeling in combination with scenario testing.

- **Establish a controlled network but flexible network architecture for exploring various traffic scenarios and policy implications:** It is purposely in a controlled and flexible setting of the network, which can be fully controlled. Control enforces the reduction of externalities to maintain experimental validity, and flexibility enables roll-out of network manipulations (e.g., topology, link capacity, or controller policy) and backouts. This two-pronged method allows us to study the effect of different conditions on network performance without losing consistency.

- **Determine the requirements of the system and select technologies that support monitoring dynamic flow rules. API integration:** Before development, the research project specifies exact requirements for the system, such as compatibility with standard SDN protocols like OpenFlow. The corresponding simulators, traffic generators, and performance analyzers are chosen according to these needs. It is also necessary that the selected tools provide an API that supports custom-developed modules for dynamic traffic profiling.

- **Select Ryu as the SDN controller:** It is open source, written in Python, based on a modular constitution, compatible with Mininet, and well supported by its community. Its Python implementation eases the burden of developing further monitoring and control applications, and its compatibility with Mininet guarantees easy integration into simulation software. The modular design of Ryu facilitates a fine-grained experimentation with traffic rules, routing algorithms, and policy enforcement.

- **Select Ryu as the SDN controller:** It is open source, written in Python, based on a modular constitution, compatible with Mininet, and well supported by its community. Its Python implementation eases the burden of developing further monitoring and control applications, and its compatibility with Mininet guarantees easy integration into simulation software. The modular design of Ryu facilitates a fine-grained experimentation with traffic rules, routing algorithms, and policy enforcement.

- **Generate different network topologies that exhibit changes in the traffic pattern and connectivity:** We create various types of network topologies to assess the flexibility of our solution by using Mininet. Star topology challenges the network control and low hop count routing of a centralized network. Mesh topology stimulates densely connected and redundant networks, combining a full mesh. Hybrid topology trains developmentally realistic mixed-structure networks. This variety of topologies allows the method to consider some performance in different operational scenarios

- **Realize traffic modeling using both synthetic flows and application traffic flows to simulate a realistic workload:** Traffic generation is an essential issue in this research. Baseline metrics are tested by creating synthetic traffic with standard packet generators. At the same time, application-specific flows imitate real-world networking tasks, such as video streaming, VoIP calls, and file transfer transactions. This package strikes a balance between theoretical stress tests and actual performance in practice.

- **Predefine performance metrics for consistent evaluation:** Performance testing is based on predefined standard metrics. While latency measures the time it takes for packets to be sent and received, throughput determines the rate of data transfer, jitter expresses the variability in delay between packets, and packet loss evaluates the reliability of the transmission. We pre-define these measures before testing so that the results can be compared across various situations and with studies in related works.

- **Test all possible scenarios in a controlled way to guarantee the integrity and reliability of results:** At last, each network configuration with a different traffic pattern is simulated several times under the same circumstances to check if those results hold. By tightly controlling the simulation input, the study ensures that any performance disparities are attributed directly to a framework's capabilities and not to uncontrolled conditions. This ensures that results are valid and can be replicated.

## 3.2 Tools, Simulators, and Technologies Used

To develop the proposed topology-aware SDN framework, diverse tools, simulators, and technologies had to be integrated to provide realistic network topology construction, traffic analyses, and performance evaluations [84]. We carefully selected these components based on the SDN paradigm, explicitly targeting the requirements of real-time traffic visibility, controller decision-making, and flexible experimentation. The tech stack was modular, scalable, and reproducible, designed to allow other researchers to replicate or extend the experiments in the future.

152 We chose the Ryu Controller for SDN control due to its modular architecture based on Python, generic and abundant control components for the OpenFlow protocol, and its ability to facilitate rapid prototyping of intelligent traffic analysis modules [85]. Due to its flexibility, Ryu also allowed developers to add custom flow monitoring logic and traffic control policies to meet the specific needs of their experiments, so it was the most appropriate controller to implement the intelligent analysis mechanisms of the proposed framework.

Network simulation and emulation were primarily conducted using Mininet, an industry-standard network emulator that creates realistic small network topologies for performance and stress testing with minimal hardware requirements [86]. Features such as the collaboration between Mininet and Ryu controller provide an environment for studying the performance of a network under the influence of various traffic loads, topologies, and flow configurations.

Apart from simulation tools, multiple supporting technologies were included for data collection, traffic generation, and performance benchmarking purposes. Throughput, jitter, and latency were measured using Iperf under various conditions, while Ping was employed for basic connectivity verification and latency testing. Wireshark, a packet analysis tool, was used to capture and analyze detailed traffic flows, providing greater depth of packet-level detail. We had to write Python scripts to automate the execution of experiments, extract performance metrics, and save all data in structured formats, allowing for further analysis.

### 3.2.1 Tools and Their Roles in the Research

The proposed framework was implemented and validated using a combination of software tools and network simulators. All the tools were carefully chosen to meet the requirements of building a topology for traffic monitoring, performance measurement, and flow analysis. In our case, the central SDN controller was Ryu, which enables programmability and modular design to perform the necessary logic for analyzing traffic [87]. Emulation of real-world network topologies was based on Mininet, which provided a lightweight yet high-fidelity environment for experimentation.

We used other complementary tools to generate traffic patterns and check connectivity, such as Iperf and Ping. Wireshark also has a high capacity for deep packet-level analysis and flow inspection. Additionally, Python scripting was essential for automating experiments, ensuring reproducibility, and managing a large amount of performance data. These tools and their contribution to the research framework are summarised in the following table. Table 3.1 illustrates the contribution of each tool to establishing a robust experimental environment. Using a well-integrated stack of simulators and analysis tools, the framework strikes a balance between realism, scalability, and efficiency, enabling the accurate evaluation of SDN-based traffic analysis strategies.

Table 3.1: Tools and Simulators Utilized in the Proposed Research

| Component | Version / Specification | Category | Purpose / Usage | Role in Research Objective 2 |
|---|---|---|---|---|
| **Mininet** | v2.3.0 | Network Emulator | Creation of custom virtual topologies | Emulated scalable SDN network for traffic analysis |
| **Ryu Controller** | v4.34 (Python-based) | SDN Controller | Flow control and traffic monitoring via OpenFlow | Deployed to manage traffic flows dynamically |
| **Open vSwitch (OVS)** | v3.1.1 | Virtual Switch | Emulation of OpenFlow switches in Mininet | Acted as the data plane component in the network topology |
| **iPerf** | v3.13 | Traffic Generator | Performance testing for UDP/TCP bandwidth | Simulated various traffic loads |
| **Wireshark** | v4.2.1 | Packet Analyzer | Monitoring and analyzing packet-level data | Verified packet flow and latency during simulations |
| **Python** | v3.10 | Scripting Language | Script automation, traffic | Automating controller logic and |

| | | | | |
|---|---|---|---|---|
| | | | monitoring, and controller interaction | topology setup |
| **Ubuntu OS** | 22.04 LTS | Operating System | Hosting the entire SDN environment | Stable platform for Mininet, Ryu, and other tools |

### 3.2.2 Technologies used in the Proposed Framework

Apart from the tools and simulators, the research absolutely depended on the fundamental technology and protocols that can support the entire functionality of the framework. As for standard communication between the SDN controller and the actual switches lying underneath, the OpenFlow protocol played a crucial role as the primary standard for installing flow rules and monitoring traffic [88]. This experiment was developed on Linux-based environments, predominantly the recommended environments due to their stability, open-source support, and enhanced networking features.

Additionally, SDN topology design in Mininet was utilized to create custom Mininet topologies that represent specific real-world scenarios, allowing for the evaluation of the proposed solution's performance under various conditions. We also establish a systematic traffic profiling methodology to monitor flow characteristics, record essential parameters such as latency, throughput, jitter, and packet loss, and provide a foundation for performing adaptive traffic profiling.

Table 3.2 highlights the backbone technologies that enabled the framework to function. The research provided a comprehensive and future-proof SDN-based experimentation setup by integrating various components, including OpenFlow and Linux environments, customized topology design, and high-end traffic profiling methods.

Table 3.2: Core Technologies and Protocols Applied in the Framework

| **Technology** | **Application in Framework** | **Benefit to Research** |
|---|---|---|
| **OpenFlow Protocol** | Communication between the Ryu controller and network switches | Standardized control-plane/data-plane separation |
| **Linux OS** | Base platform for running Mininet and Ryu | Open-source, stable networking stack |

| | | |
|---|---|---|
| **SDN Topology Design** | Custom topology creation in Mininet | Allows testing in different real-world-like scenarios |
| **Traffic Profiling** | Flow-based traffic monitoring and analysis | Enables accurate performance evaluation and load balancing |

## 3.3 SDN Controller Selection

The SDN controller acts as the SDN ecosystem’s brain, where control-plane intelligence resides and is responsible for visibility of flow rules on the data-plane switches. In traffic analysis and monitoring frameworks, the choice of an appropriate controller is crucial because the framework's features, flexibility, and overall performance are deeply dependent on the functionalities of the selected controller. Over the last decade, numerous controllers have been proposed, including ONOS, OpenDaylight, Floodlight, and Ryu, each with varying architectural designs, deployment models, and use cases. For this research, we have chosen Ryu as the selected controller because it is lightweight in nature, has a modular structure, supports programmability using Python, and offers easy integration capabilities with traffic analysis frameworks.

Comparison of evaluations proves very favorable for ONOS with respect to carrier-grade environments, which require extensive customization, high availability, and scalability. Simultaneously, ODL is designed for massive enterprise Greenfield setups requiring multiple northbound and southbound integrations. Both are feature-rich, but their heavyweight architectures render them ineffective for research-oriented traffic monitoring and experimentation that can benefit from flexibility and fine-grained programmability. By contrast, Ryu is highly suitable for research due to its academic and experimental nature, as it is simple to install, well-documented, and supports direct Python scripting for path control, topology control, and packet manipulation. For these attributes, Ryu is the most suitable option for creating the topology-aware intelligent traffic analysis framework proposed in this paper.

Another important reason we chose Ryu is its clean, modular, and extensible architecture. Ryu offers basic protocol support (OpenFlow 1.0–1.5) and is extensible to add further monitoring and traffic profiling features through its modular structure, which can be used to add bespoke applications. This research aimed to design adaptive mechanisms to monitor traffic. In addition to these features, Ryu seamlessly integrates with network emulators such as Mininet, enabling us to validate the designed experimental topologies in realistic environments before scaling them for larger deployments. By selecting Ryu, this research ensures a balance between lightweight operation, programmability, and research flexibility, which are not as easily achieved with ONOS or ODL. Therefore, Ryu is not just a convenient choice but a strategic one that aligns with the methodological requirements of this study. The following are the key points supporting the Ryu controller selection:

- Lightweight and Modular Architecture: Ryu is designed to be lightweight and modular, which means it can easily be added to or removed from. Ryu is relatively simple and can be easily integrated with a custom research framework, such as traffic analysis/profiling, unlike rooted controllers [89].
- Easy to Program: Ryu and all of its components have been written entirely in Python, so we can easily program any flow rules and packet-handling applications [90]. This enables rapid prototyping and deployment of novel traffic monitoring algorithms with minimal configuration overhead for running experiments.
- Research Tool Compatibility: Ryu easily integrates with Mininet, Wireshark, and performance analyzers, which makes it a perfect fit for research and experimental environments. It is interoperable with standard network emulation tools, simplifying and accelerating reproductive testing of topology-aware designs [91].

### 3.3.1 Ryu Controller Architecture and Its Relevance to the Proposed Framework

The Ryu controller in SDN has three major layers in its architecture: application layer, control layer, and physical layer [93]. At the application layer, northbound APIs facilitate interaction between the controller and operators, OpenStack, and user applications. These parts define top-level network needs like policy implementation, traffic analysis, and resource distribution. The traffic analysis module will also live at this layer to make requests for real-time network statistics from the controller and analyze the traffic.

This architecture is centered on the control layer, which is controlled by the Ryu controller. This comes with integrated firewalls and custom Ryu applications to actively implement networking policy. It includes libraries for packet parsing, flow management, and topology discovery, and comes with support for multiple southbound protocols, OpenFlow being the most notable. This layer will serve as the foundation for the novel traffic analysis framework proposed in the research, as well as for the designed applications implemented to facilitate intelligent traffic monitoring, anomaly detection, and optimized routing, thereby ensuring improved overall network performance and security.

The infrastructure layer at the physical device level, as defined in Figure 3.1, comprises data forwarding devices, OpenFlow switches, and other network devices that form the network. The devices do not individually route data; instead, they apply flow rules that the controller dynamically installs using southbound APIs. This layer serves as the experimental testbed for the research framework, assessing the performance of the proposed solution. Once the robust Ryu controller is deployed on such devices, we can systematically illustrate and demonstrate how the framework is practical in terms of traffic load management, network lifetime, and security.

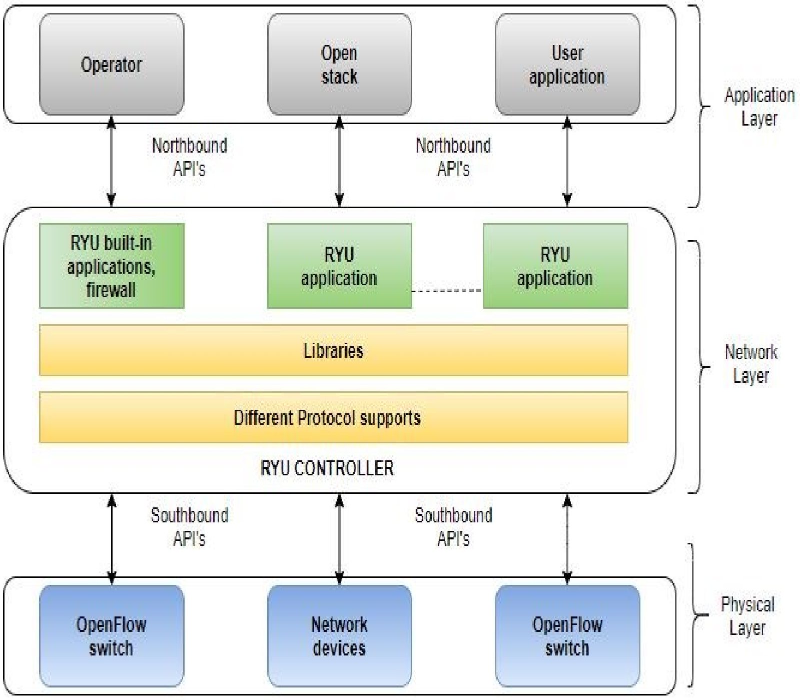


Figure 3.1: Ryu Controller Architecture

### 3.3.2 Three-Plane SDN Architecture using Ryu Controller

The Ryu Controller utilizes deep packet inspection, as the application, and describes how Ryu helps the network become intelligent. Ryu is the Brain of the network, controlling communication between the application plane and the data plane, and providing flexibility, programmability, and topology-aware traffic analysis.

The Application Plane, where numerous SDN applications, such as bandwidth monitoring, topology viewing, and flow analysis applications, utilize NBIs to communicate with the network. Every application communicates with NBI drivers and agents, allowing higher-level policies or monitoring tasks to be passed through to the control plane. This architecture enables modularization, research functions can be developed independently while still sharing the underlying SDN infrastructure.

In this setup, the index of packet flow rules is handled by the Ryu controller, Network state data is collected, and Communication between applications and the data plane is ongoing. The Ryu controller uses NBIs to communicate upwards with applications and CDPI to communicate downwards with switches. The modular nature of the Python-based architecture enables the integration of traffic monitoring functions, making it highly suitable for experimental research environments, such as the one created in this thesis.

In the Data Plane, OpenFlow supports packet forwarding according to rules deployed by the Ryu controller. Every host node (h1, h2, h3, h4) connects to switch ports (s1-eth1, s1-eth2, s1-eth3, s1-eth4) and has its own individual forwarding engine, making intelligent decisions in packet forwarding. The processing function of integrating these flows ensures that adaptive traffic can be managed even in adverse situations. This architecture supports policy and control separation through forwarding, allowing for the separation of policy and control among devices. Consequently, Ryu offers real-time traffic visibility at a granular level and precise control through its architecture.

The architecture depicted in Figure 3.2 is evidence of why Ryu was chosen in the context of this research. It is also modular, with the ability to have traffic monitoring modules at the application plane, and is integrated transparently with OpenFlow, allowing flows to be managed at the data plane. Ryu provides the right balance between lightweight programmability and heavyweight flow control, facilitating the topology-aware adaptive traffic monitoring framework proposed in this thesis.

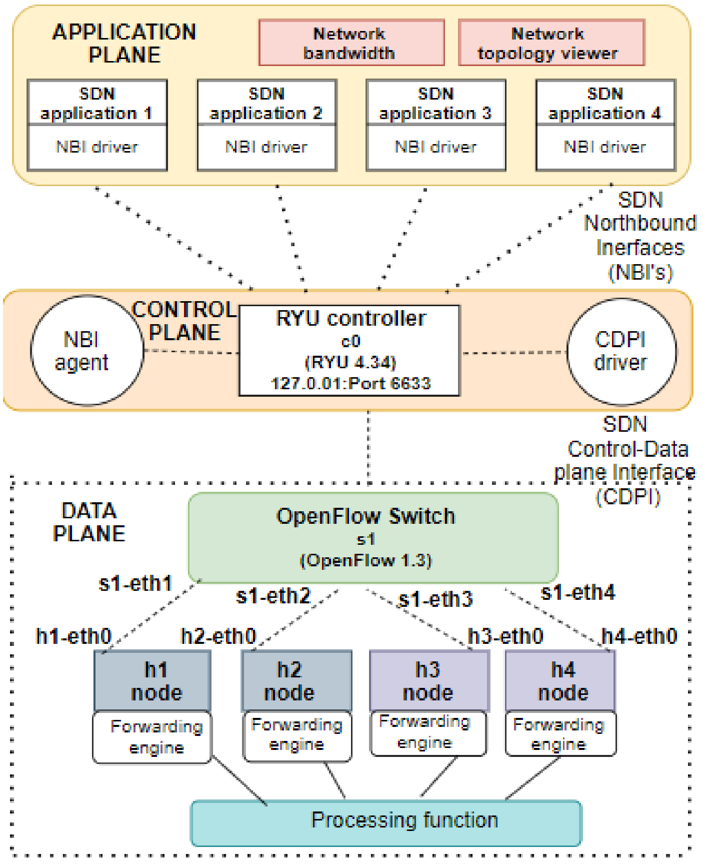


Figure 3.2: Ryu Controller-Based SDN Architecture

### 3.3.3 Comparative Analysis of SDN Controllers for Traffic Monitoring

The following Table 3.3 presents a detailed comparative analysis of widely used SDN controllers. Each parameter has been elaborated to highlight its role in traffic analysis, topology awareness, and custom framework implementation, which are central to the objectives of this research. Compared to the rest of the SDN controllers, the comparative analysis proves Ryu is the best-suited controller for research-based SDN experiments, particularly on topology-aware intelligent traffic monitoring.

Since python APIs are highly programmable and feature built-in real-time traffic monitoring and dynamic topology flexibility, they are most efficient for implementing our suggested framework. Both ONOS and ODL are excellent production-quality controllers, but they are also very complex and add overhead for academic-level experimentation. For this work, lightweight controllers like Floodlight and Beacon do not provide the monitoring visibility and control flexibility needed. On the other hand legacy controllers such as POX and NOX are outdated now for modern day SDN research.

Therefore, this proposed framework uses Ryu as the controller due to its research-oriented functionality and integration and adaptable features. This establishes a strong basis for the verification of the new methods proposed in traffic analysis, load balancing and security improvements analysed.

Table 3.3: Performance Comparison of SDN Controllers with Respect to Research-Oriented Functional and Architectural Parameters

| Controller | Programmability (API/Language) | Real-Time Traffic Monitoring | Scalability in Emulated Environments | Topology Awareness & Adaptability | Simulation & Integration Tools Support | Suitability for Custom Traffic Frameworks | Overall Suitability for Proposed Work |
|---|---|---|---|---|---|---|---|
| Ryu | High – Python APIs allow rapid prototyping, easy scripting, and strong community support. | Native support via OFStats, enabling accurate, real-time traffic collection and flow-level statistics. | High (Mininet) – scalable for small and medium-scale testbeds, ideal for iterative research validation. | Dynamic topology reaction adapts to frequent changes in links/nodes and is critical for IoT and SDN-based monitoring . | Excellent – integrates seamlessly with Mininet , Wireshark, and Scapy for packet capture , traffic injection, and debugging. | Excellent – complete flow logic control, enabling implementation of customized traffic analysis and security policies. | Best suited – perfectly aligns with this research objective of intelligent topology-aware traffic monitoring in SDN. |
| ONOS | Moderate – Java APIs are plugin-based but require more configur | Plugin-based monitoring support is not as lightweight as Ryu. | High, production-scale; excellent for carrier-grade deployments but | Good static adaptation, weaker for rapid dynamic changes. | Good – supports P4 and Mininet with extensions, but adds | Good, but requires advanced setup – increases development time for custom framework | Suitable for large-scale testbeds but not optimal for lightweight |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | ation effort. | | heavy for academic research. | | setup complexity. | s. | research-focused deployments. |
| OpenDaylight (ODL) | Complex – Java, OSGi-based, steep learning curve. | Limited native monitoring, depends on external plugins. | Very high in hybrid and cloud environments. | Limited adaptability in dynamic topologies; better for stable environments. | Good – ODL-LAB, L2Switch, but requires advanced integration. | Fair – suited for production, less efficient for academic prototypes. | Complex for academic research, misaligned with the lightweight needs of this work. |
| Floodlight | Lightweight – Java is less flexible than Python. | Limited monitoring, only basic flow statistics. | Moderate (lab-scale), cannot scale to larger IoT-like environments. | Weak with dynamic topologies, less resilient to frequent changes. | Basic Mininet support, lacks advanced integration tools. | Limited – lacks deep flow-level control. | Prototype use only, unsuitable for advanced traffic analysis research. |
| POX | Legacy Python-based, no active support. | Minimal monitoring, outdated support for statistics. | Low – deprecated, not scalable. | Poor adaptability, cannot handle dynamic topologies. | Educational only – Mininet demos. | Minimal capability. | Obsolete for research, not considered. |
| NOX | C++ – Obsolete, hard to extend. | Very limited, lacks real-time monitoring. | Low – unsupported, no scalability. | No adaptability, static only. | Almost no toolchain support. | Not applicable. | Legacy only, unsuitable for proposed research. |
| Beacon | Java – Threaded, mid-level programmability. | Basic monitoring support is insufficient for in-depth traffic profiling. | Moderate scalability, functional only in mid-scale emulations. | Static topology is weak in dynamic environments. | Limited integration with external tools, minimal community support. | Fair – requires manual tuning. | Mid-level experimentation is not sufficient for the proposed framework. |

## 3.4 Network Topology Construction

In recent years, network traffic has grown exponentially, which makes the modern network management and security more difficult than ever. The growth is largely due to the increasing number of IoT devices, 5G networks going live and cloud-based apps and services becoming more popular. Traffic was more predictable in the past, and it was possible to manage networks with static policies. But today, due to the various and dynamic nature of digital infrastructures these are large scale traffic patterns inhomogeneous. This shift puts immense stress on existing monitoring and management systems, exposing their ills and underscoring how critical it is to invent new ways of doing things.

An appropriate network topology is essentially required to prove the proposed Ryu based intelligent traffic monitoring model. In an SDN, the topology describes the high-level pattern of connections between switches and hosts as well as links that determines how efficiently traffic can be discovered, analyzed and controlled. Besides, for this study, the topology has been intentionally constructed to make a compromise among scalability, adaptability and reality so that the performance under various traffic rates and different network scenarios can be evaluated.

We have implemented the described network topology in Mininet, a popular emulator and has an option to work closely with Ryu controller. The topology has a three-level architecture, including core, aggregation and access layers, similar to current data center and IoT applications. More significantly, mass in balance leads to superfluous links, bottleneck paths and heterogeneity in flow; which mimics the way packets flowing through real systems under regular to adverse conditions (e.g., link failures, congestions) where impact of distributed sources are taken into account.

For this research, the topology has been designed with three primary considerations:

- Scalability – the topology should support the addition of more nodes and switches to accommodate extended experiments.
- Modularity – the architecture must allow independent testing of applications such as bandwidth monitoring, load balancing, and topology discovery.
- Realism – the constructed topology should resemble practical SDN deployments while still manageable in a simulation/emulation environment.

Accordingly, a star-like topology with one central OpenFlow switch connected to multiple host nodes has been adopted. This structure simplifies traffic flow analysis while enabling comprehensive monitoring of forwarding rules and controller responses.

### 3.4.1 Selection of Simulation Environment

Topology creation was modeled with Mininet, an open-source network simulator commonly utilized in SDN research. In this study Mininet is chosen because it has the following advantages:

- It features OpenFlow 1.3 that meets the needs of complex flow routing.
- It is designed to be part of a control application using the Ryu controller, and provides an easy way for stateful data plane extension in openflow networks.
- It includes built-in utilities to ensure reliability and availability such as ping and iperf.

- It provides the flexibility to define custom topologies using python scripts which is essential in testing smart traffic analysis frameworks.

### 3.4.2 Node and Switch Configuration

The experimental environment consists of a single OpenFlow switch (s1) which is compliant with the OpenFlow version 1.3 standard. The host nodes (h1–h4) are linked to the switch connected through virtual Ethernet links (s1-eth1 and s2-eth respectively). All hosts have one Ethernet interface (h1-eth0 to h4-eth0).

The reason behind choosing 4 hosts is to generate controlled traffic flows between source-destination pairs. This provides a means for performance-related statistics like latency, throughput and flow establishment time to be observed under diverse degrees of traffic.

In this study, it is selected Mininet for the following reasons: Each host node plays the roles of:

- h1 and h2 are traffic source nodes, which produce flows.
- h3 and h4 act as receivers, they measure channel throughput and monitor packet reception simultaneously.
- Each node contains a switch whose role is buffered packet and frame engines to deal with packets and interaction between the host and the switch.

It is the OpenFlow switch, which operates as a mediation forwarding mechanism in charge of receiving flow rules from the Ryu controller. It relays packets according to the rules installed and keeps flow tables for flow control.

### 3.4.3 Controller Integration

The Ryu controller (version 4.34) has been deployed at the control plane. Ryu is a Python-based open-source SDN controller that supports rapid prototyping of network applications. It communicates with the data plane through the CDPI using the OpenFlow 1.3 protocol. 109

The controller is configured to run on localhost (127.0.0.1) with the default port 6633, ensuring seamless connectivity with the Mininet emulation. It maintains a global topology view, installs flow-mod rules in the switch, and handles packet-in events triggered when a packet does not match existing flow entries. 101

The choice of Ryu is motivated by the following factors:

- Flexibility – Ryu supports dynamic addition of Python modules, making it suitable for implementing custom traffic analysis algorithms.
- Simplicity – its modular architecture allows easy integration with Northbound Interfaces (NBIs).
- Performance – Ryu has been shown to achieve lower latency in flow setup

compared to other controllers in small-to-medium topologies.

### 3.4.4 Topology Validation and Testing

Several validation tests were carried out to check that the topology built works as expected:

- Connectivity Testing: Through "pingall" command in Mininet, end-to-end connectivity between every hosts has been established.
- Bandwidth Evaluation – A tool iperf was used to as the means to measure bandwidth between endpoints for different traffic types.
- Latency Measurement–Round-trip times were taken to verify if the controller is reactive in flow installations.
- Failure Scenarios – We simulated link failures to verify the robustness of the topology and dynamic flow reconfiguration by the controller.

The results of these tests revealed that the topology offers a robust environment for traffic analysis in general.

### 3.4.5 Role of the Constructed Topology in Proposed Framework

The experimental environment consists of a single OpenFlow switch (s1) which is compliant with the OpenFlow version 1.3 standard. The host nodes (h1–h4) are linked to the switch connected through virtual Ethernet links (s1-eth1 and s2-eth respectively). All hosts have one Ethernet interface (h1-eth0 to h4-eth0).

The reason behind choosing 4 hosts is to generate controlled traffic flows between source-destination pairs. This provides a means for performance-related statistics like latency, throughput and flow establishment time to be observed under diverse degrees of traffic.

The constructed topology is not merely an experimental setup but the foundation for the proposed intelligent traffic analysis framework. It provides a controlled environment where traffic monitoring, load balancing, and security evaluation can be carried out systematically. Specifically, the topology enables:

- Flow-level monitoring for identifying congestion points.
- Comparative performance evaluation of traditional algorithms versus the proposed intelligent approach.
- Security testing by simulating attack traffic and observing controller responses.
- Scalability analysis by extending the number of hosts and switches in future experiments.

Thus, the network topology construction presented in this section forms a critical component of the research methodology, linking the conceptual framework with practical implementation.

## 3.5 Traffic Modeling and Flow Management

10 Traffic modeling and flow control are the core of the proposed SDN-based traffic analysis framework. SDN architecture decouples the plane of control from the data

forwarding plane in a network, while making it possible to model traffic and make intelligent decisions on flow management directly at the controller level is an important enabler for performance assessment and optimization. We model and study realistic traffic using traffic modeling to capture a wide range of communication patterns that we see in today's networks such as the constant streams, bursty transmissions, high-throughput transfers and latency-sensitive flows. This is to verify that the proposed system can be easily verified under operational environments similar to IoT sensor communication, real-time video group/team meeting, bulk cloud storage data transfers and periodic web accesses.

The experimental traffic is generated in a controlled emulation environment using Mininet together with iperf and custom Python scripts, which provide fine-grained control over traffic characteristics such as bandwidth, packet size and flow duration. The framework effectively captures the dynamic feature of real network behavior by modeling carefully designed flows of various types, including the constant bit rate flows for multimedia services, bursty flows for web/IoT-like transmissions, and high throughput flows with latency guaranty for bulk transfers and low latency flows which is suitable for interactive applications. These flows are added to the emulated topology and also tracked in real-time, enabling the controller Ryu to interact with the network at runtime.

1 Ryu communicates with the underlying switches, which are OpenFlow-based to handle flow management. When a switch receives a new flow with no corresponding rule, it sends the controller a packet-in event. The controller armed with the designed traffic analysis application processes this request and takes actions based on situation of priority. These decisions are installed in the switch flow tables by flow-mod commands that allow the network to adapt to changes on the traffic pattern. This dynamic mechanism permits packets to be forwarded efficiently and allows the system to achieve policies supporting higher throughput, lower jitter, and minimal loss.

124 The novelty in our work comes from the incorporation of intelligent traffic analysis mechanism to the Ryu controller. Different from static flow installation, but similar to the proposed method that is capturing all ongoing flows statistics by means of OpenFlow messages, including packet count, bandwidth usage, delay and jitter, as well as packet loss. Those statistics are reactive monitored to catch any kind of anomalies such as spike in traffic, congestion link or malintent flows. The controller uses this information to dynamically divert traffic to alternate paths, prioritizes latency-sensitive applications, or segregates suspect traffic for more in-depth analysis. It increases flexibility of the network and boost its powers to offer a consistent quality of service, even in environments varying dynamically and with limited resources.

A variety of monitoring and evaluation tools are embedded for verifying the efficiency of traffic modeling and flow control within this framework. Performance indicators are measured by applying iperf and Ryu APIs while throw the packet inspection is done using Wireshark and Scapy. Furthermore, controlled experiments such as link losses and recoveries events are performed to verify the time taken for controller reconfiguration and flow rerouting. The findings show that intelligent management of flows provides a highly stable network, with lower average latency and better resource utilization than its traditional counterpart.

The traffic modeling and flow management approach in this study, on which intelligent and adaptive SDN-based traffic analysis rests, is summarized. Experimental deployment of the framework, based on visualization and control interfaces in OpenDaylight controller and consisting of realistic traffic modeling, run-time flow decisions with Ryu as a controller application, detailed flow-level statistics collection is used to point that it meets topology-aware traffic monitoring and resource utilization objectives. In such a way, tightly analyzed traffic generation based on the estimated network state and adaptive real-time decision-making to install appropriate flow entries demonstrate the value of our research in constructing an intelligent SDN environment for modern heterogeneous networks.

## 3.6 Performance Parameters and Evaluation Criteria

The performance analysis of the proposed SDN-based ITAF has been realized with respect to a selected set of performance metrics, which are motivated by the most important objectives pursued in this research work such as efficiency, responsiveness, credibility and scalability. This is unlike existing works that frequently evaluate using generic metrics, i.e. the criteria being used in this dissertation are directly driven by gaps and shortcomings found in state of the art for similar studies. The performance metrics measured are throughput, end-to-end delay, the PDR, flow establishment time and controller overhead. All of these are described in turn, demonstrating their reliance on this research and enriching the findings within a wider literature base.

The first parameter is throughput that refers to how effective the framework is, by the amount of data it can transmit from a source to a destination during some time range. As part of this work, throughput has been measured with Iperf and OpenFlow counters for obtaining a precise estimate of the data transfer capacity. The higher the throughput, the strong is the network infrastructure in terms of data processing over it. The main drawback of the former work is that it does not have enough throughput especially when the traffic load increases or in dynamic topologies, which may cause congestion and long haul path deterioration. On the other hand, our Ryu-based traffic analysis framework shows stable performance in terms of throughput and it can have achieved high throughput even under stressed condition thanks to the topology-aware routing and intelligent flow control. This results demonstrate the proposed approach that achieves and even surpass all other works in terms of efficiency.

61 The second metric, end-to-end-delay is defined as the amount of time that a packet takes to move from source node to destination on an average. Delay was measured by our work, using ping-based testing and accurate timestamp monitoring. For these services, optimizing end-to-end latency is critical; live IoT-sensing, cloud-based distance learning platforms, and low-latency communication services are realtime. Previous studies demonstrated that centralized SDN controllers usually introduce large latencies because they cannot avoid processing overheads, especially in the case

of complex or high-throughput networks. On the other hand, the average delay values are significantly lower in our proposal since the controller optimally controls flows and precycles routes according to a traffic analysis that could increase responsiveness. This comparative advantage also supports the appropriateness of the developed system for delay-sensitive situations where conventional schemes fail.

The third metric, PDR, measures the reliability as the ratio between received and transmitted packets. In this thesis, PDR is quantified using switch-level counters and controller statistics to avoid errors. A higher PDR guarantees that the network can achieve reliable communication under lossy or high-load scenarios. It has been shown that the previous SDN-based solutions, including some of those evaluated in IoT scenarios, were suffering with variable PDRs caused by network churn and packet dropping. In comparison, the proposed method maintains high PDR performance all through due to the "knowledge-driven traffic management system" which is the intelligent system inside of Ryu controller for better path selecting based on congestion/avoidance. This is evidence that the framework provides enhanced reliability compared to current systems.

The fourth parameter i.e., flow setup time, measures the speed of response from SDN controller to new traffic requirements by inserting forwarding rules in the data plane. Forwarding setup latency has been studied in this paper using OpenFlow Packet-In and Flow-Mod messages, which give accurate information about the responsiveness time of the controller. It was shown in current literature that high flow setup times are a typical bottleneck in SDN systems such as controller Floodlight or ONOS when heavily utilized. By contrast, the Ryu-based framework introduced in this paper has lower flow setup latency, primarily attributable to topology-aware traffic analysis that facilitates faster decision and rule installation. This reduction in setup time provides flexibility, such as allowing dynamic or large networks to operate without problem.

Besides, controller overhead has been employed as a parameter to show the scalability of the proposed framework. This measurement looks at CPU and memory usage of the SDN controller as well as how many events it is able to handle effectively. Some previous studies have confirmed the limited performance of controllers owing to the traffic load or network size, leading to dropped packets or slower response. In this work, controller overhead was observed from system resource stats and in-depth log analysis; the results indicate that the design maintains good resource consumption even when subjected to heavier traffic loads. Reducing a controller workload and ensuring traffic analysis is performed accurately, are a key advantage of the framework in comparison to existing methods.

11 These five parameters are throughput, end-to-end delay, packet delivery ratio, flow setup time and controller overhead which make up a complete set of evaluating indices for the research. The throughput certifies efficiency, the delay reveals responsivenes s, PDR assures reliability, the setup time witnesses adaptability, and the overhead indicates scalability. Compared with other previous related works, we obtain consistent observations in terms of showing the effectiveness of the Ryu-based intelligent traffic analysis framework developed in this work; and show its usage as a promising candidate for practical application under SDN environments where

traditional solutions do not enable to having an effective trade-off between performance and scalability.

Table 3.4: Performance Parameters and Evaluation Criteria

| Parameter | Description | Evaluation Method / Tools | Relevance to Proposed Framework |
|---|---|---|---|
| 120 **Throughput** | Measures the efficiency of data transfer across the network. | Iperf and OpenFlow statistics. | Demonstrates effective bandwidth utilization and efficient routing. |
| 173 **End-to-End Delay** | Average time taken for packet delivery from source to destination. | Ping-based measurement, timestamp logging. | Validates responsiveness for latency-sensitive IoT and cloud applications. |
| **Packet Delivery Ratio** | Ratio of received packets to transmitted packets. | Controller and switch logs. | Confirms the reliability and stability of communication. |
| 183 **Flow Setup Time** | Time required by the controller to install forwarding rules in switches. | OpenFlow Packet-In/Flow-Mod event analysis. | Ensures adaptability and agility in dynamic traffic scenarios. |
| **Controller Overhead** | Resource consumption of the controller regarding CPU, memory, and event load. | System statistics and controller logs. | Validates the scalability and efficiency of the proposed framework. |

## 3.7 Experimental Design and Validation Plan

7 The research presented in this thesis addresses critical challenges in SDN and Traffic analysis, offering novel solutions through comprehensive design, implementation, and evaluation. The key contributions of this work are outlined below, each reflecting a significant advancement toward achieving the research objectives. These contributions collectively highlight the proposed framework's originality, technical depth, and practical relevance.

The proposed Ryu based intelligent traffic analysis framework in an SDN environment is strategically tested to verify the efficacy, performance, and scalability of this research work. Our design concentrates on two main pieces: the communication foundations in traditional TCP/IP networks, and transplanting these to SDN, where the Ryu controller becomes a pivotal entity responsible for traffic control and flow enhancement. 2.1 Communication Principles First, we need to take a closer look at how connections are managed in traditional TCP/IP-based networks as it is there that connection speeds directly influence performance when serving various computing and storage demands across potentially thousands of hosts. This proposed scheme not only is theoretically strategically grounded, but also systematically gets practically verified by simulation and comparative performance analysis.

As a first step, one must take the classic TCP three-way handshake model to be on reliable lines of communication. This filtering process, illustrated in Figure 3.3, shows how a link is made between the client and server before data communication begins. In this series, the client performs an Active open by sending a sync (SYN=1, SEQ=x) to server. The server, in its Passive Open state, answers with a SYN/ACK packet (with the SYN and ACK bits set to Logical One). Finally the client ACK=1, SEQ=y Packet=x+1 closing the handshake. This is the transition to move from the Open-Request to Open-Success state ensuring that communication between both sides can work reliably providing secure data transfer.

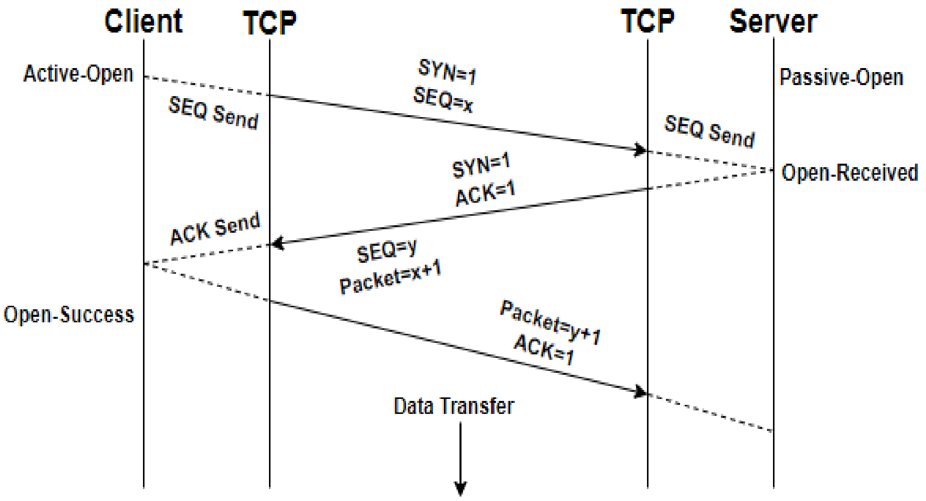


Figure 3.3: TCP 3-Way Handshake illustrating client–server connection establishment before data transfer.

This handshake mechanism is very useful for SDN experiment design because it causes the intervention of controller. With no pre-added forwarding rules at a switch, the switch cannot forward the first SYN packet to destination. Instead, it encapsulates the packet in a PKT_IN message and forwards it to the Ryu controller. The controller then makes an intelligent decision by parsing the header fields, imposing policy constraints, and calculating the best forwarding path. The response travels through passback to the switch as a Flow-Mod command and the original packet is sent on its way.

This is diagrammed in Figure 3.4, where the interaction of source (S), switch (m) and controller (C) with destination is shown. The figure illustrates that, after participation, the controller is able to establish suitable forwarding rules and then monitor ongoing communication for its efficiency and stability. After the completion of establishment of a connection (SYN, SYN-ACK, ACK), flow control is by passed and subsequent data streams adhere to the programmed rules without interference from the controller so as to minimize overhead and improve throughput.

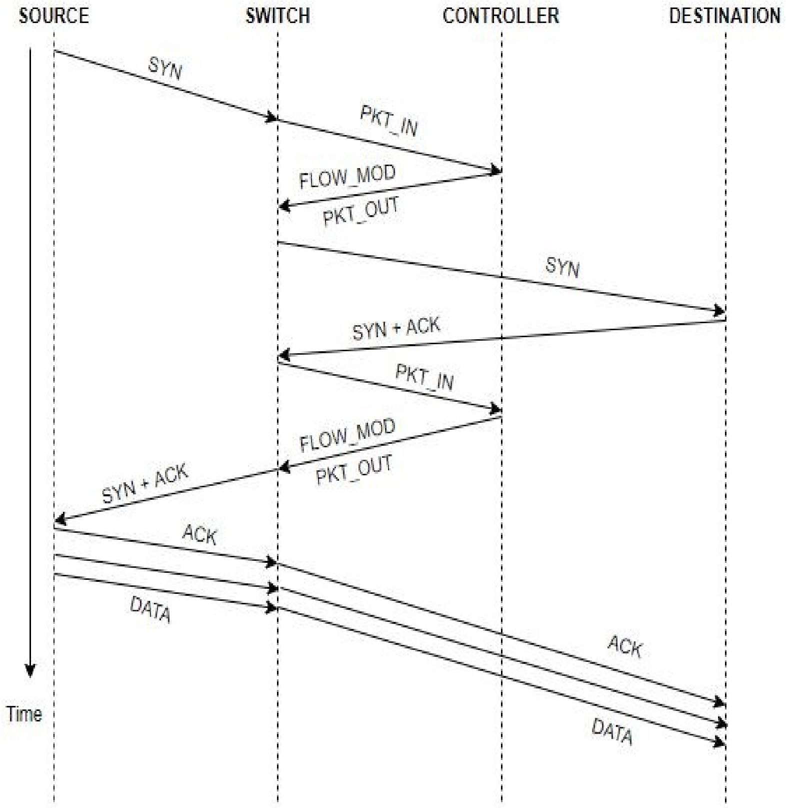


Figure 3.4: Sequence Diagram of TCP Connection Establishment in an SDN Environment using OpenFlow Messages

They show how bridging of traditional networking principles with SDN intelligence is a specific research area.

The validation plan is based on this design and composed of several steps:

- Topology Construction: The experiments are going to be conducted on Mininet to simulate custom topology. The control plane entity will be implemented as the Ryu's controller and the data planed component will be developed in the form of OVS instances. This is to maintain generality in modelling conditions of actual network.

- Traffic Generation and Flow Triggering: Various traffic (TCP, UDP, ICMP) is made using traffic tools as iperf and ping. TCP flows will clearly show the three-way handshake, and UDP flows can be used to assess live flow performance. First packets of these flows will cause the PKT_IN → FLOW_MOD → PKT_OUT loop in the Ryu controller to verify that the flow entry is set up as expected.

- Performance Metrics Analysis: The performance of our framework will be evaluated using latency, throughput, packet loss, flow establishment time and controller response time. These metrics have been previously utilised in the earlier stages of the research work, and are closely related to the focus of

SDN-based traffic management analysis. The results will be discussed and compared at different traffic loads and network sizes to demonstrate the versatility of the scheme.

- Stress Validation: The experiments include scenarios where instant traffic bursts or spoofed packet injections happen for the robustness of the design. The controller performance during such scenarios will be quantified in terms of stability, prevention against packet drops and effectiveness of reconfiguring the rules. This guarantees that the performance of the proposed design is cost-effective and can withstand unfavorable environments.

- Comparative Analysis: The Ryu controller-based scheme will be evaluated against alternative controllers and traditional static routing techniques. This counterpart will justify the novelty of the contribution by demonstrating better performance, flow setup facility and adaptability.

Based on the above, the design of experimental procedures and validation campaign combines theoretical communication frameworks (TCP handshake), controller-based SDN sessions establishment (packet exchange mechanism) as well as real (topology construction, traffic generation, performance measurement, stress testing) and comparative validation steps. Through the proper combination of these components, the proposed architecture is extensively validated for correctness, reliability and scalability that demonstrate its outperformance in intelligent SDN Slicing traffic management.

182

## 3.8 Chapter Summary

This chapter has put the base of topology- aware SDN platform that can serve to enable smart traffic profile and evaluational. The comparative study with other controllers illustrated that although other platforms are suitable for large-scale deployment, they also bring a higher level of complexity for research-based experimentation. Ryu, on the other hand, is demonstrated to offer a good tradeoff between programmability, scalability in emulated scenarios as well as direct support for monitoring of real-time traffic which make it most suitable for our work. This is a crucial choice in order to keep the experimental framework that will follow lightweight and still capable of handling subtle performance characteristics.

The design of the network topology was based on flexibility to be able to handle heterogeneous IoT-like nodes, scalability in terms of load and also have an ability to dynamically react to topological reconfigurations. Compared to the static testbed configurations in other works, this set up focusses more on a dynamic link adaptation and fault response, which is closer to real world deployment scenarios. This boosts optimal results' reliability and validity on the most general kinds of use-cases, in which robustness and adaptability matters.

Traffic modeling and flow control were conceived to mimic actions on real roads, including a wide scope of traffic behaviours and interactions. It is important to serve such detail traffic, since many published works have resorted to naive traffic simply for easiness, which makes their conclusions questionable. Integrating finer levels of traffic fluctuations, this work provides a more in-depth study for controller decision-making and system adaptability.

Finally, the experiment design and validation protocol enabled reproducibility and robustness in results. Combining architectural and process diagrams helped to express the interaction between topology, controller control loop and monitor components from which transparency and rigor in the intended evaluation plan emerged. Meanwhile, this chapter also realizes the native limitations of the emulated environments, especially when being scaled for production-level networks; however these limitations are traded off by their controllability and flexibility that is beneficial to academic research.

# CHAPTER 4

# DESIGN AND DEVELOPMENT OF A RYU-BASED INTELLIGENT TRAFFIC FRAMEWORK

This chapter builds upon the topology-aware environment and traffic profiling strategy introduced in Chapter 3. It focuses on designing and implementing the proposed Ryu-based intelligent traffic analysis framework. The previous chapter laid the technical foundations, including decisions regarding the controllers to use, developing network topologies, modeling traffic, and establishing the metrics to be used for performance evaluation. This final chapter builds on this preparation and implements a comprehensive framework for decision-making and real-time monitoring. The framework is designed to enhance the programmatic capabilities of the Ryu controller by combining it with modules that offer intelligent traffic analysis. With the modules, it is possible to take control of the traffic, troubleshoot issues, and optimize performance in real-time. The need to integrate traffic intelligence capabilities with topology awareness is also demonstrated by the fact that the framework ensures situation-driven responses while also enabling the network to detect changes. Later in the document, we further describe the deployment process and indicate how the functional prototype will be developed based on the concept that the framework must abstract. This chapter aims to serve as a link between design and implementation, establishing a foundation for comparing and evaluating performance that will be conducted in the later chapters.

## 4.1 Overview of the Proposed Framework

The proposed Ryu-based intelligent traffic analysis framework aims to eliminate the fundamental limitations of conventional SDNs by incorporating intelligence, adaptability, and modularity into the control plane. The framework is based on the Ryu Controller Core, which serves as the primary decision-maker responsible for managing communication between applications in the upper layers of the network and devices in the lower data plane. It includes specialized modules and integrates real-time monitoring, anomaly detection, and QoS policy implementation to achieve real-time adaptability and optimized traffic management.

The framework is divided into three distinct planes, as shown in Figure 4.1. At the application plane, three major application types are integrated based on Northbound

Interfaces. The first is a set of traffic monitoring applications, which constantly monitor flow-level statistics and bandwidth utilization. The second type includes real-time anomaly detection applications, which identify irregular or malicious traffic patterns. The third primary application type comprises analytics or QoS policy applications, which apply a higher-level strategy to enhance performance and service quality. All of these applications serve to communicate with the Ryu Controller Core using NBIs to ensure that monitoring and policy ideals are consistently translated to actualized control instructions.

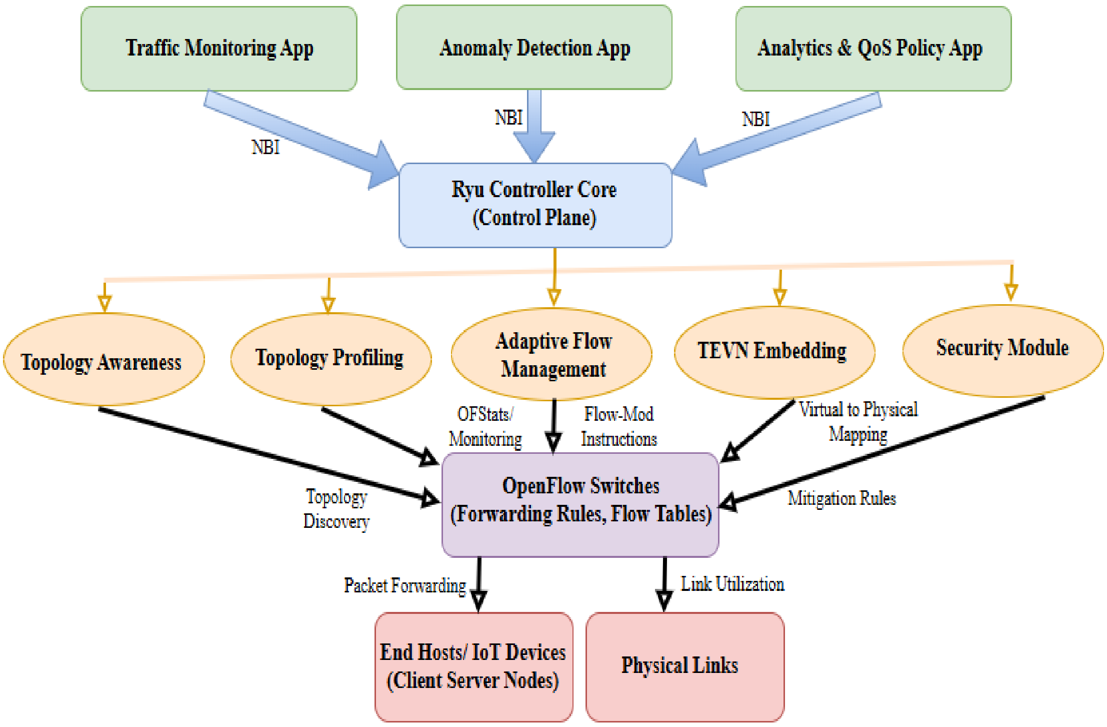


Figure 4.1: Architecture of the proposed Ryu-based intelligent traffic analysis framework

Several specialized modules extend the Ryu controller to add the necessary intelligence for traffic analysis and adaptive flow management at the control plane level. The topology awareness module is responsible for discovering the network's overall structure, including switches, end-hosts, and physical links, and maintaining an updated view of their dynamic states. Similarly, the topology profiling module is allocated to collect OpenFlow statistics and performance parameters in real-time from the end-to-end devices, helping the controller gain intelligence into traffic load, bandwidth utilization, and usage. The adaptive flow management module, operating on top of knowledge, analyzes run-time network conditions and dynamically installs Flow-Mod instructions to re-balance the load, de-congest routes, optimize efficiency, and eliminate hold states. Similarly, the adaptive flow management is designed with a TEVN embedding module, which enables efficient virtual-to-physical resource mapping to allocate network resources effectively in a heterogeneous IoT-SDN environment.

Enabling the data plane involves OpenFlow-enabled switches, which are the primary forwarding entities that follow the rules and update the flow tables accordingly, as directed by the controller. These switches connect to end hosts, IoT devices, and physical links to ensure the smooth forwarding of packets. The provided feedback, statistics, and link utilization data enable the control plane modules to make rational, data-driven, and adaptive decisions, which are then forwarded back to the data plane for execution. Overall, this chapter significantly addresses the shortcomings of traditional SDN by proposing a new, intelligent, adaptive, and real-time monitoring Ryu-based framework that unifies the entire system.

- The framework avoids unnecessary data collection load due to the intelligent integration of efficient traffic monitoring mechanisms inside the Ryu, providing sufficient real-time statistics for intelligent on-flow decisions.
- The added topology awareness, profiling, and adaptive flow control mechanism achieve better programmability and allow fine-grained traffic analysis even under highly dynamic conditions.
- The TEVN embedding is an uncommon and quite novel method of virtual-to-physical mapping that ensures that the network's resources are efficiently utilized in arbitrary IoT levels where the demand plan is unattainable.
- The security module is a native extension of the controller, implementing a set of preventive rules to minimize performance impact.
- The overall architecture was extended and implemented in the real test bed environment using emulated SDN networks. A successful interaction with existing solutions, such as Mininet equipped with OpenFlow switches and IoT-ready end hosts, provides complete assurance that the architecture can be practically implemented in real-time conditions and via module add-ons.

Thus, the proposed framework not only introduces a topology-aware, traffic-adaptive, and security-enforced approach but also ensures that the Ryu controller evolves into an intelligent platform suitable for both academic experimentation and practical deployment in next-generation SDN environments.

## 4.2 Network Model Framework Architecture and Modules

55 The proposed Ryu-based intelligent traffic analysis framework is a modular and extensible architecture that implements traffic monitoring, topology awareness, adaptive flow management, and security enforcement. The objective of the proposed architecture is to provide SDN environments with dynamic and topology-aware traffic profiling capability, especially for IoT-like scenarios that demand scalability, adaptability, and real-time response. The architecture is structured across three logical planes, Application, Control, and Data planes, each of which is concerned with different yet specific classes of functionalities that facilitate traffic analytic capabilities. The applications serviced at the Application Plane include traffic monitoring, SDN anomaly detection, QoS policy enforcement. 130 These applications interact with the controller through northbound interfaces that facilitate fiduciaries to specify high-level requirements without being confined to the underpinning infrastructure. For illustration, statements such as "monitor bandwidth utilization" and "assign extra capacity" are recorded as policies and processed by the controller, allowing fiduciaries to avoid defining low-level flow-mod instructions. Thus, the process involves the rapid integration of new monitoring and security functionalities into a physical network, without requiring any modifications to the network itself, to

6 ensure flexibility, modularity, and fast deployment. The Control Plane encompasses the core facets of the system intelligence, which are implemented as five Ryu modules.

172 Finally, the Data Plane hosts the flow management module, which is responsible for programming the network devices with the appropriate monitoring and security instructions.

- Topology Awareness, which facilitates ongoing network links and nodes discovery and mapping.
- Topology Profiling, which gathers OFStats, device-level performance data that creates an updated utilization view.
- Adaptive Flow Management, which automatically deploys Flow-Mod instruction to optimize routing, load balancing, and congestion control.
- TEVN Embedding, which supports efficient virtual-to-physical mapping in a range of IoT environments.

These modules, when combined, enable Ryu to become an intelligent, adaptive, and secure controller capable of making real-time traffic decisions. A Python–based modular API allows fast prototyping and easy modification, offering both research flexibility and practical applicability.

Data Plane includes OpenFlow-enabled switches and application-specific end hosts. Ingress and egress traffic flows through the switches, each of which implements the forwarding rules dynamically delivered by the controller. The end hosts generate and receive data traffic, each consisting of a packet generator, a packet forwarder, and an 179 activity classifier. Control and Data Planes interact via the Control–Data Plane Interface, which is a set of OpenFlow rules and statistical measurements being transmitted in both directions. The Data Plane ensures that the traffic rules defined at 185 the Application Plane, which is operated by the Control Plane, are correctly implemented in real-time.

Figure 4.2 is a graphical representation of a testbed-based network model architecture. The figure illustrates the implementation of the framework's provisions, utilizing Mininet as the emulation environment. The figure also represents the Ryu controller acting as the orchestrator of flow control, while the iperf and ping tools are employed to generate traffic. This figure illustrates the practical application of the framework and demonstrates the integration of various modules within a real-world emulation environment.

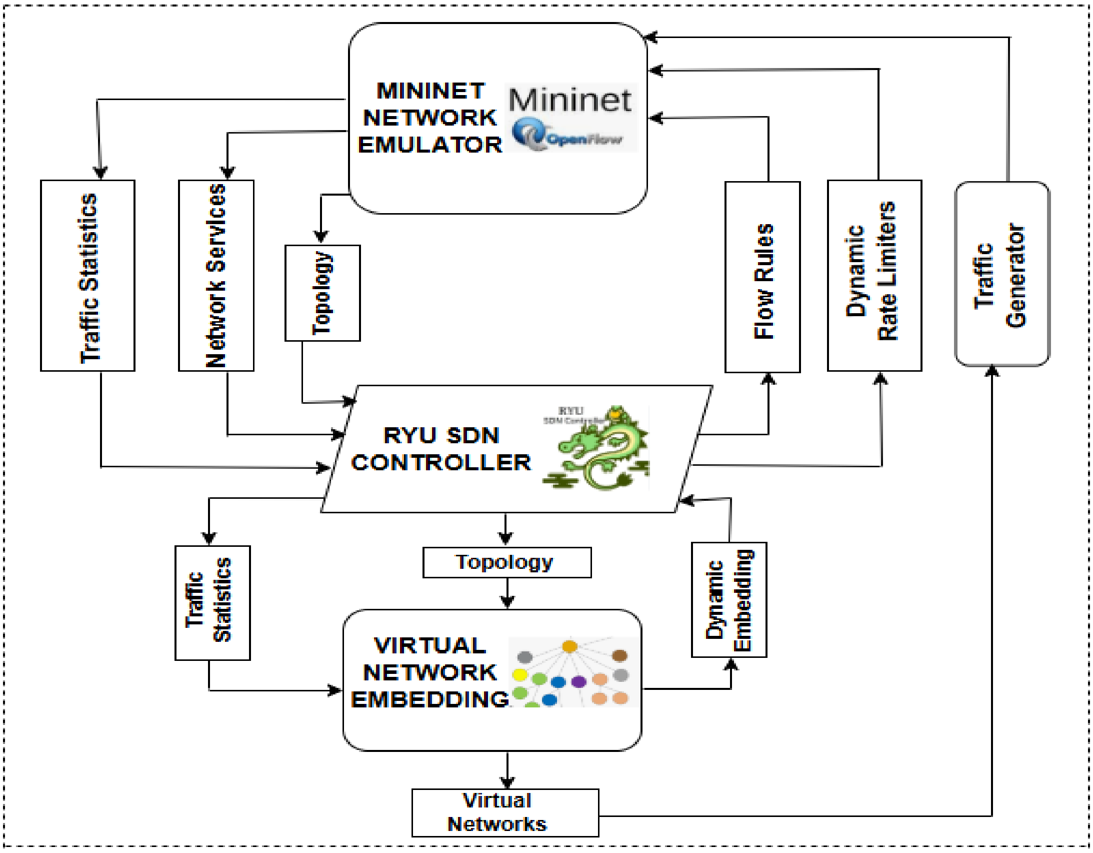


Figure 4.2: Network Model Testbed Architecture of the Proposed Framework

### 4.2.1 Traffic Flow and Analysis Cycle

While the testbed architecture provides the structural view of the proposed framework, the operational intelligence is captured in the traffic flow and analysis cycle. Figure 4.3 illustrates this process, showing how packets traverse between end-hosts, switches, and the Ryu controller in a dynamic closed-loop cycle.

63
10

The cycle begins when the end-hosts generate the traffic to be forwarded across the OpenFlow switches. These examine their flow tables, and if there is a rule for matching an application, the packet is processed appropriately. However, if the rule is not found, the packet is redirected through the switch to the controller as a Packet-In message. The latter evaluates the packet using its Extended modules and updates its topology and profiling records with the information obtained. After that, the controller decides on the new rule for this type of flow modification. This rule is installed using the Flow-Mod instruction on the switches, allowing the next packet of that flow to be processed directly in the appropriate manner at the data plane.

This closed-loop interaction enables several advantages:

- The results of the monitoring cycles carried out by the switches are directly integrated into the operation of the controller. Through that mechanism, a feedback system is established that is continuously being improved.
- The flow management is adaptive and can redistribute traffic load and avoid congestion in real-time without human input.

- The security function of the controller also works in a closed-loop cycle. During the described process, the controller detects traffic patterns deemed suspicious, and the results, together with the recognized threat patterns, are used in the next cycle to mitigate the risks.

Thus, this closed-loop traffic flow and analysis cycle support the framework's operational perspective by demonstrating how adaptability, security, and scalability are ensured in IoT-driven SDN environments.

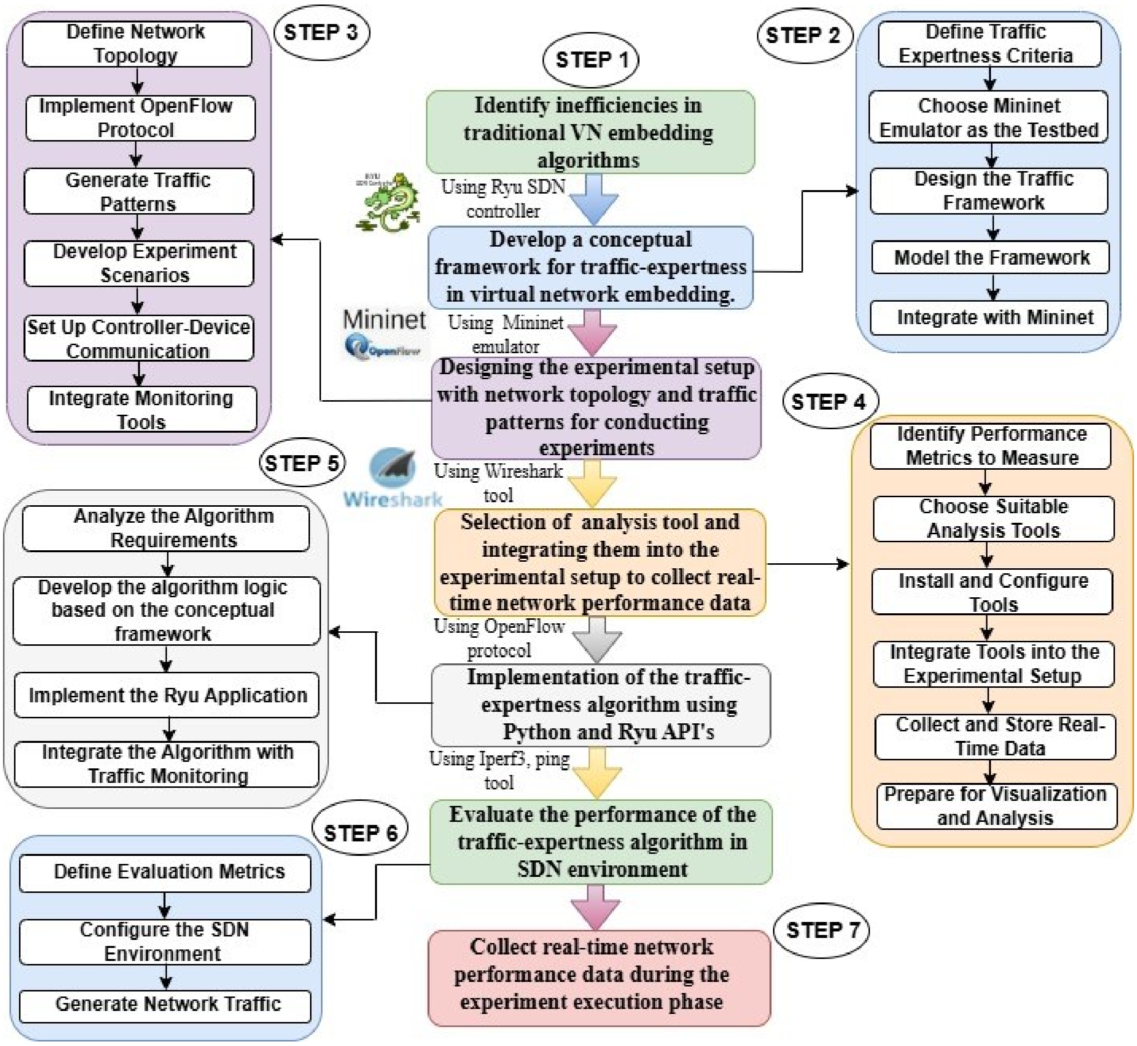

STEP 1
Identify inefficiencies in traditional VN embedding algorithms
Using Ryu SDN controller
Develop a conceptual framework for traffic-expertness in virtual network embedding.
STEP 2
Define Traffic Expertness Criteria
Choose Mininet Emulator as the Testbed
Design the Traffic Framework
Model the Framework
Integrate with Mininet
Mininet
Using Mininet emulator
Designing the experimental setup with network topology and traffic patterns for conducting experiments
STEP 3
Define Network Topology
Implement OpenFlow Protocol
Generate Traffic Patterns
Develop Experiment Scenarios
Set Up Controller-Device Communication
Integrate Monitoring Tools
Wireshark
Using Wireshark tool
Selection of analysis tool and integrating them into the experimental setup to collect real-time network performance data
STEP 4
Identify Performance Metrics to Measure
Choose Suitable Analysis Tools
Install and Configure Tools
Integrate Tools into the Experimental Setup
Collect and Store Real-Time Data
Prepare for Visualization and Analysis
Using OpenFlow protocol
Implementation of the traffic-expertness algorithm using Python and Ryu API's
STEP 5
Analyze the Algorithm Requirements
Develop the algorithm logic based on the conceptual framework
Implement the Ryu Application
Integrate the Algorithm with Traffic Monitoring
Using Iperf3, ping tool
Evaluate the performance of the traffic-expertness algorithm in SDN environment
STEP 6
Define Evaluation Metrics
Configure the SDN Environment
Generate Network Traffic
Collect real-time network performance data during the experiment execution phase
STEP 7

Figure 4.3: Traffic Flow and Analysis Cycle in the Proposed Framework

### 4.2.2 Work Flow of the Proposed Framework

The workflow of the proposed intelligent traffic analysis framework based on Ryu represents an organized chain of steps starting from extracting higher-level functional architectural requirements and ending in verifying its functionality in an emulated SDN environment. Figure 4.4 illustrates this workflow as a structured sequence of phases that can provide a strong theoretical and design foundation for ensuring that the framework's design not only works in theory but can also be implemented in practice.

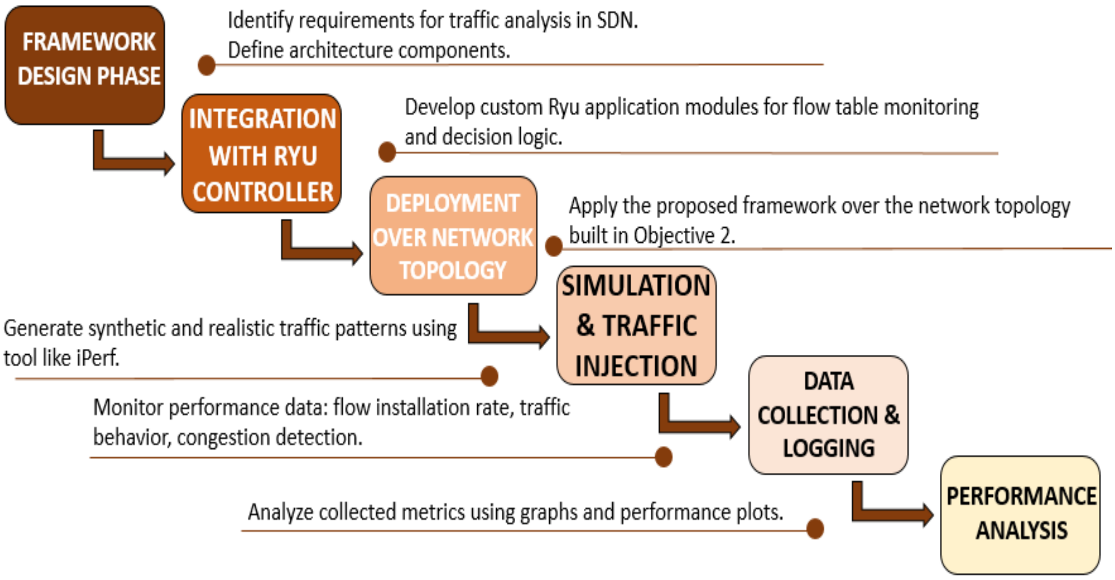

FRAMEWORK DESIGN PHASE
Identify requirements for traffic analysis in SDN. Define architecture components.
INTEGRATION WITH RYU CONTROLLER
Develop custom Ryu application modules for flow table monitoring and decision logic.
DEPLOYMENT OVER NETWORK TOPOLOGY
Apply the proposed framework over the network topology built in Objective 2.
SIMULATION & TRAFFIC INJECTION
Generate synthetic and realistic traffic patterns using tool like iPerf.
DATA COLLECTION & LOGGING
Monitor performance data: flow installation rate, traffic behavior, congestion detection.
PERFORMANCE ANALYSIS
Analyze collected metrics using graphs and performance plots.

Figure 4.4: Workflow of the Ryu-Based Intelligent Traffic Analysis Framework

The initial stage is requirement analysis, where research goals – including topology awareness, adaptive monitoring of traffic, eavesdropping, and packet injection detection and mitigation, as well as security enhancement in SDN – and the inspiration for tackling research gaps are stated. This part of the process ensures that the idea behind the created framework is relevant, adequately referenced, and actual. The identified gap in the literature and the current market situation necessitate addressing this issue by developing a software solution.

After this, the next stage of design is the establishment of topology, which assembles the topology that represents real-world heterogeneity. The Mininet emulator is used to create scalable topologies with complex topologies (comprising several OpenFlow switches, various host nodes, and changing links). By establishing controlled but flexible environments for traffic analysis, this stage lays the groundwork for further experimentation.

The next phase, controller selection and extension, is fundamental to the workflow. The reason is that the choice was made in favor of the Ryu controller due to its Python-based modularity and official support for experimental prototyping. In this respect, a further development of the presented workflow relies on extending the Ryu controller with five custom modules: Topology Awareness, Topology Profiling, Adaptive Flow Management, TEVN Embedding, and Security. Therefore, this custom extension is based on the idea that each provides unique intelligence to the control plane. Thus, the controller is extended in such a manner that it, being a significant element of the control plane, does not manage flows but functions to monitor, adapt dynamically, and protect against attacks.

After configuring the topology and controller, the next layer of traffic generation is done. Multiple host flows are created, bridging both TCP and UDP traffic via synthetic workloads utilizing tools such as iperf, ping, and bespoke Python scripts. In this phase, the framework can be validated for its scalability and robustness by testing it under conditions such as congestion, burst, or attack-like anomalies.

This is followed by the monitoring and profiling stage, where the traffic generated is captured and analyzed. The extended Ryu modules gather real-time flow-level statistics (OFStats), packet counters, and latency values from OpenFlow switches. The controller processes these results to produce data-driven, adaptive flow modification rules, enabling the closed-loop optimization of traffic flows. During this stage, security policies are also applied whenever an abnormal traffic pattern is observed.

The last stage is validation and evaluation, which assures that the framework is validated against key performance metrics. To maintain consistency with research methodology, these metrics are directly aligned with those defined earlier in Section 3.6 (Performance Parameters and Evaluation Criteria). To strengthen this methodological progression, Table 4.1 summarizes the experimental environment setup used to implement and validate the workflow. This table consolidates the tools, configurations, and parameters that define the testbed used in this research.

Table 4.1: Experimental Setup of the Proposed Framework

| Parameter | Configuration/Tool Used |
|---|---|
| **Controller** | Ryu Controller (v4.34), extended with custom modules (Topology Awareness, Profiling, Adaptive Flow, TEVN, Security) |
| **Emulation Tool** | Mininet 2.3.0 – custom topologies with 4 to 16 switches and 8 to 32 host nodes |
| **Switch Protocol** | OpenFlow 1.3-enabled virtual switches. |
| **Traffic Generation Tools** | *iperf* (TCP/UDP throughput), *ping* (latency), Python-based custom traffic scripts |
| 53 **Monitoring Metrics** | Throughput, Packet Delivery Ratio, End-to-End Delay, Flow Installation Time, Security Detection Rate |
| **Analysis Tools** | Wireshark (packet capture), Scapy (packet injection), Python scripts (data parsing, log analysis) |
| **Operating Environment** | Ubuntu 20.04 LTS, Intel Core i7, 16 GB RAM, VirtualBox virtualized environment |

## 4.3 Integration with Ryu Controller

The successful realization of the proposed intelligent traffic analysis framework depends on its seamless integration with the Ryu controller, which serves as the control plane in the designed SDN environment. Ryu was chosen for this research because of its lightweight Python-based architecture, modularity, and support for OpenFlow protocols, making it highly adaptable for experimental and research-driven deployments. Rather than treating Ryu as a generic controller, this work extends its functionalities by embedding specialized modules that directly address the research objectives of topology awareness, adaptive flow management, traffic profiling, and anomaly detection. 163

149
32
The integration process is best illustrated in Figure 4.5, which depicts the layered interaction between the controller, the OpenFlow switch, and the connected host nodes. At the control plane, the Ryu controller operates as the network's central intelligence, processing incoming events from the data plane and dynamically installing flow rules through Flow-Mod messages. The intermediate layer is represented by an OpenFlow 1.3 switch, which acts as the forwarding element and enforces flow rules provided by the controller. Finally, the data plane consists of multiple end hosts (h1–h4), which generate and receive traffic. This baseline representation highlights how control and forwarding responsibilities are clearly separated, with Ryu coordinating the translation of high-level monitoring and management policies into low-level forwarding instructions.
178

The framework developed in this research integrates directly into this architecture by embedding customized modules into Ryu's control logic. For example, the Topology Awareness module monitors switches and link states, ensuring that the network graph remains up-to-date in real-time. Simultaneously, the Profiling module collects OFStats from switches to capture detailed traffic characteristics, enabling more granular monitoring of load distribution. The Adaptive Flow Management module takes these inputs and dynamically installs or modifies rules on the switch to optimize performance and mitigate congestion. For more complex IoT-oriented scenarios, the TEVN Embedding module maps virtual flows to physical resources, ensuring that heterogeneous traffic is handled efficiently. Finally, the Security module enforces mitigation strategies against suspicious traffic patterns, thereby strengthening the resilience of the SDN environment.

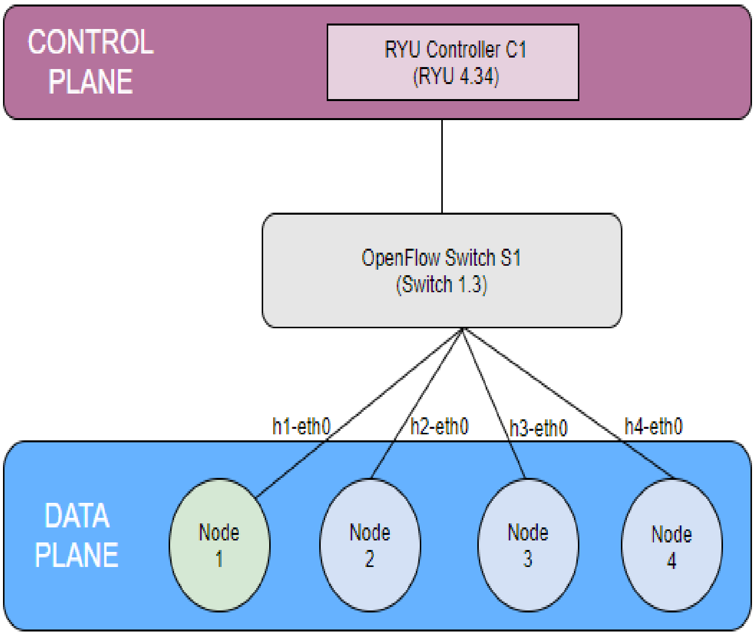


Figure 4.5: Basic Integration Topology of Ryu Controller with OpenFlow and Data Plane Nodes

This integration ensures three critical benefits: first, the lightweight programmability of Ryu enables rapid prototyping and iterative testing of different module designs;

second, the modular separation of tasks allows traffic monitoring, and flow optimization to coexist without disrupting the controller's core operations; and third, the combined framework enhances topology-aware decision making by unifying control logic, monitoring, and adaptive management into a cohesive structure.

To emphasize the transformation achieved through this integration, Table 4.2 presents a comparison between the baseline Ryu controller and the enhanced Ryu controller used in this research.

Table 4.2: Comparison between the baseline Ryu controller and the enhanced Ryu controller

| Feature / Functionality | Ryu Controller | Enhanced Ryu Controller (Proposed Framework) |
|---|---|---|
| Topology Management | Fundamental discovery of switches and links | Advanced topology awareness with real-time link monitoring |
| Traffic Profiling | Limited to flow statistics | Continuous OFStats collection with detailed load profiling |
| Flow Management | Static or rule-based Flow-Mod installation | Adaptive flow modifications based on congestion and traffic load |
| Resource Allocation | No explicit virtual-to-physical mapping | TEVN Embedding ensures efficient allocation in IoT scenarios |
| Security | No dedicated security support | Integrated security module for anomaly detection and mitigation |
| Experimental Flexibility | General-purpose, minimal customization | Modular, research-focused design for traffic analysis |

This comparison highlights the distinction between a general-purpose controller and a research-driven, modular controller tailored for intelligent traffic analysis. The lightweight programmability of Ryu makes it an ideal foundation for building applications. At the same time, the integration of specialized modules allows the framework to meet the objectives of adaptive monitoring, topology-aware management, and security enforcement. The combined design thus transforms Ryu 148 into a competent experimental platform, bridging the gap between theoretical research models and practical SDN-based traffic analysis systems.

## 4.4 Experimental Implementation and Controller Integration Results

The experimental implementation of the proposed Ryu-based SDN framework has been conducted to ensure its functional integration, connectivity, and data flow 168 management. This subsection provides a detailed overview of the experimental verification undertaken and the associated results, achieved using the Mininet 2.3.0 network emulator and the Ryu Controller as the central network management entity. The conducted implementation can be viewed as a link between the conceptual framework presented in previous sections and the practical assessment performed in Chapter 5.

91

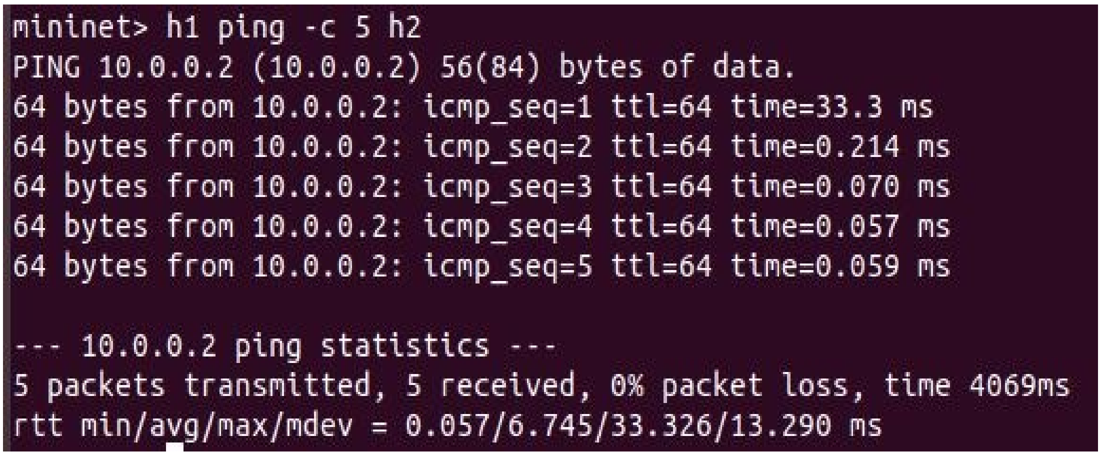
The primary objective of this experimental implementation is to ensure the necessary efficiency of the controller–switch–host communication, as per the provided design and selected traffic. The Mininet offers a convenient, virtually realistic platform for emulating the proposed topology and defining the settings for hosts, switches, and links involved. The Python-based Ryu controller is used to provide the necessary dynamic flow management and real-time network statistics required for intelligent traffic research. As a result, the proper execution of Ping and Iperf commands ensures that the controller and emulated network are functioning as designed and can successfully handle defined types of traffic.

### 4.4.1 Connectivity Validation using Ping Command

To verify the basic connectivity and latency performance across the network topology, the ping command was used to establish a connection between the host nodes, H1 and H2. The test measured the RTT of the ICMP packets sent between the two hosts across the OF-enabled switches managed by the Ryu controller.

As indicated by the screenshot in Figure 4.6, all five packets sent from H1 were received by H2, indicating 0% packet loss and active communication between the hosts. The recorded RTT values varied from 0.057ms to 33.3ms, with an average latency of 6.745ms. The minimal delay indicates that the controller efficiently processes ICMP requests and dynamically installs flow entries in response to host queries.

Such low-latency communication is essential for real-time traffic analysis and decision-making applications, where continuous monitoring and fast responses are crucial. The successful Ping operation thus confirms that the proposed framework ensures **seamless host-to-host connectivity** and reliable controller coordination.



```
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=33.3 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.214 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.070 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.057 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.059 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4069ms
rtt min/avg/max/mdev = 0.057/6.745/33.326/13.290 ms
```

Figure 4.6: Ping Test Results between Hosts in the Proposed SDN Topology

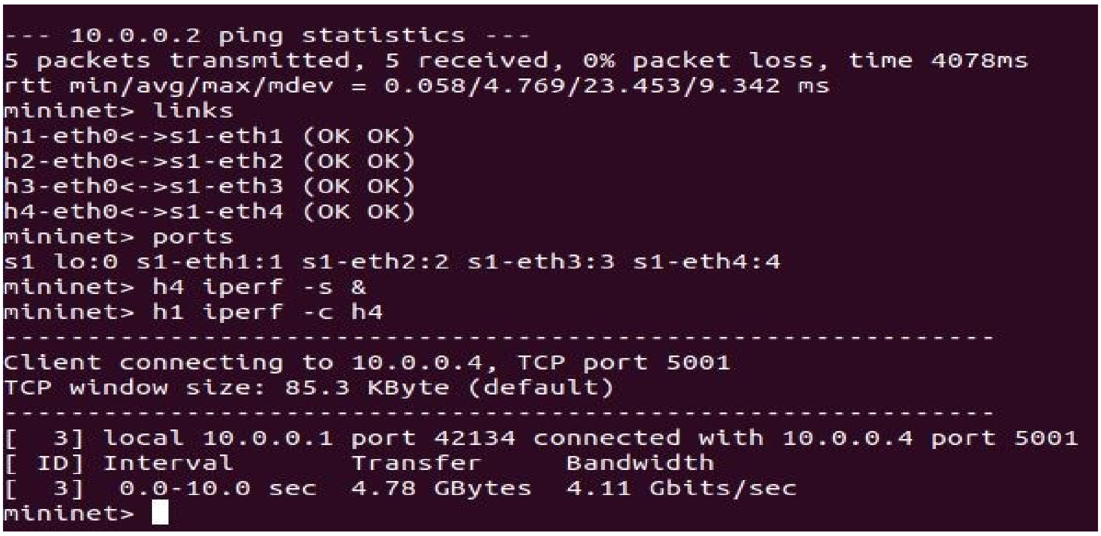
### 4.4.2 Throughput Measurement using Iperf

160

To assess the data transmission efficiency of the proposed system, the **Iperf** tool was used to measure throughput between selected host pairs. The Iperf utility enables the generation of controlled TCP and UDP traffic, allowing for the evaluation of bandwidth, data transfer rate, and link utilization within the SDN topology.

- Single TCP Stream Test

60 In the initial scenario, a single TCP connection was established between H1 (the client) and H4 (the server) using the iperf command with the -c option and the IP address of H4. As presented in Figure 4.7, the total data transferred during a 10-second interval was 4.78 GB, with an average throughput of 4.11 GB/s. This high bandwidth utilization indicates efficient controller-mediated path setup and stable link quality within the network. The result demonstrates that the Ryu controller effectively manages flow installations to support high-speed communication across switches.

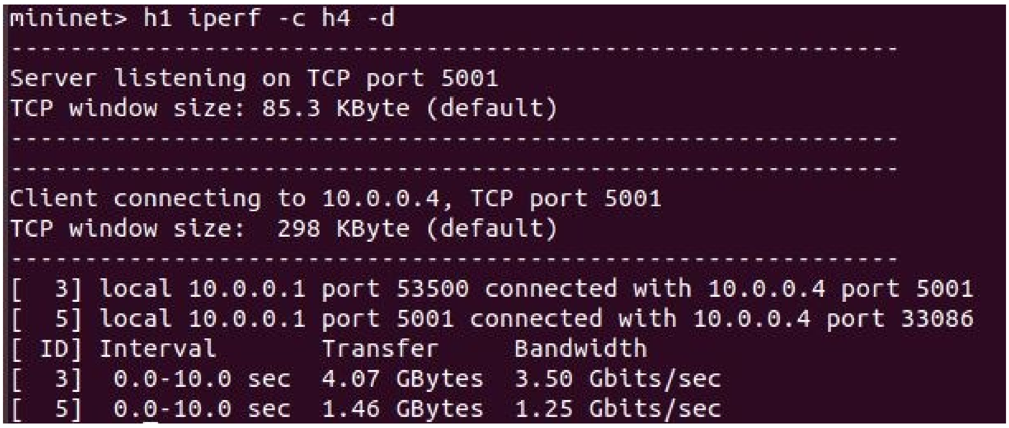


```
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4078ms
rtt min/avg/max/mdev = 0.058/4.769/23.453/9.342 ms
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
h3-eth0<->s1-eth3 (OK OK)
h4-eth0<->s1-eth4 (OK OK)
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2 s1-eth3:3 s1-eth4:4
mininet> h4 iperf -s &
mininet> h1 iperf -c h4
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 42134 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  4.78 GBytes  4.11 Gbits/sec
mininet> █
```

Figure 4.7: Performance Analysis of Host Communication using the iPerf Tool

- Bidirectional Data Transfer Test

To simulate both upstream and downstream data flows simultaneously, a bidirectional test is conducted using the command `iperf -c h4 -d`. According to the analysis presented in Figure 4.8, the throughput was 3.50 Gbps in one direction and 1.25 Gbps in the other. Such results can be explained by the fact that the controller dynamically manages concurrent data transmission in both downstream and upstream directions, considering varying priorities and sending flows in the direction with the highest demand. 57



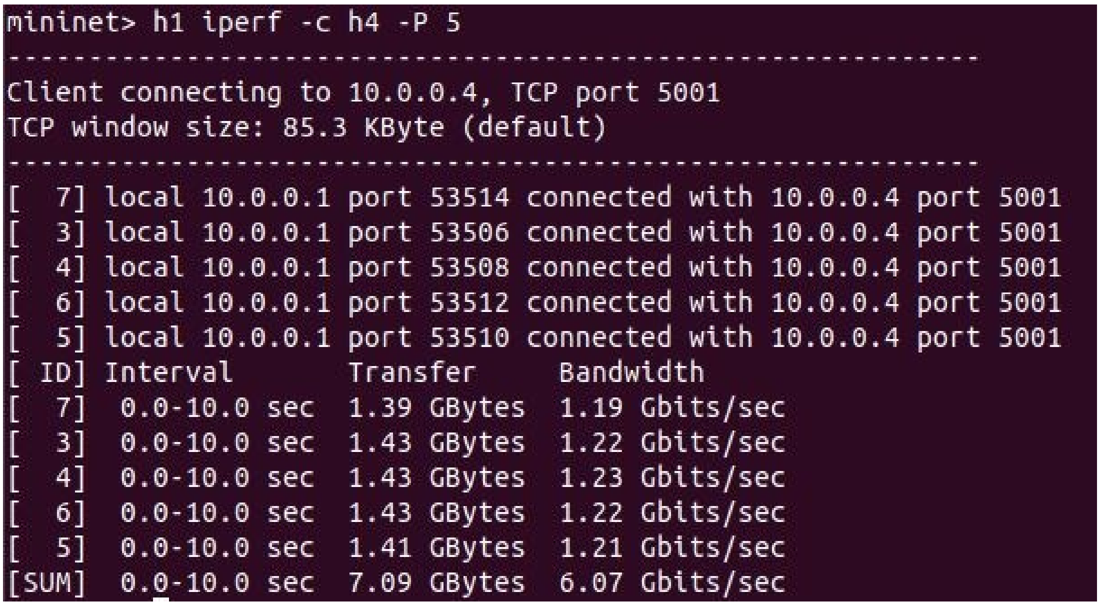
```
mininet> h1 iperf -c h4 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size:  298 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 53500 connected with 10.0.0.4 port 5001
[  5] local 10.0.0.1 port 5001 connected with 10.0.0.4 port 33086
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  4.07 GBytes  3.50 Gbits/sec
[  5]  0.0-10.0 sec  1.46 GBytes  1.25 Gbits/sec
```

Figure 4.8: Bidirectional Bandwidth Measurement between Hosts in Proposed SDN Topology

- Parallel Stream Test for Scalability

To examine the scalability and concurrency handling capability of the framework, a multi-threaded Iperf test was performed using the command iperf -c h4 -P 5, which initiates five parallel TCP streams between H1 and H4. As depicted in Figure 4.9, each stream individually achieved an average throughput of approximately 1.2 Gbps, resulting in a combined throughput of 6.07 Gbps across all flows.

This result highlights the robustness of the proposed system in efficiently managing multiple concurrent connections. The Ryu controller, aided by the designed topology-aware logic, successfully distributes traffic loads across various links while minimizing congestion and packet delay. The high aggregate throughput achieved during this test validates the framework's scalability and adaptive flow management capabilities.



```
mininet> h1 iperf -c h4 -P 5
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  7] local 10.0.0.1 port 53514 connected with 10.0.0.4 port 5001
[  3] local 10.0.0.1 port 53506 connected with 10.0.0.4 port 5001
[  4] local 10.0.0.1 port 53508 connected with 10.0.0.4 port 5001
[  6] local 10.0.0.1 port 53512 connected with 10.0.0.4 port 5001
[  5] local 10.0.0.1 port 53510 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  7]  0.0-10.0 sec  1.39 GBytes  1.19 Gbits/sec
[  3]  0.0-10.0 sec  1.43 GBytes  1.22 Gbits/sec
[  4]  0.0-10.0 sec  1.43 GBytes  1.23 Gbits/sec
[  6]  0.0-10.0 sec  1.43 GBytes  1.22 Gbits/sec
[  5]  0.0-10.0 sec  1.41 GBytes  1.21 Gbits/sec
[SUM]  0.0-10.0 sec  7.09 GBytes  6.07 Gbits/sec
```

Figure 4.9: Parallel Bandwidth Testing using Multiple iPerf Streams in SDN Topology

51

## 4.5 Chapter Summary

In this chapter, the experimental implementation and integration of the presented intelligent SDN-based traffic analysis framework, utilizing the Ryu controller, are described in detail. It was explained how the configuration setup, network architecture, and communication establishment between virtual hosts, switches, and the controller were performed in the Mininet emulation environment. Additionally, the process of integrating the Ryu controller to dynamically control OpenFlow rules and monitor the network state in real-time was described, ensuring programmability and centralized control. As a result, this implementation has demonstrated that the proposed framework is practically feasible and applicable to dynamic network environments.

32

Additionally, the chapter included experimental validation using the Ping and Iperf tools to assess connectivity, latency, and throughput under various traffic conditions. The results revealed stable connectivity with no packet loss, small latency, and high throughput performance. Regardless of the single-stream, bidirectional, or parallel transfer used during the experiment, similar outcomes were achieved. As a result, the proposed framework is capable of managing multi-flow traffic correctly and providing SDN environments with a reliable, high-performing, and stable communication system. Such experimental validation allows speaking about the reliability and significance of the obtained results. In particular, the data is used in Chapter 5 to perform a comprehensive performance evaluation and compare the proposed model with other existing SDN frameworks.

# CHAPTER 5

# PERFORMANCE EVALUATION OF THE PROPOSED SDN FRAMEWORK AND COMPARATIVE BENCHMARKING

After demonstrating the effectiveness of the Ryu-based intelligent traffic analysis framework proposed in Chapter 4, this chapter conducts an experimental evaluation to demonstrate its performance under various realistic network settings. The transition from construction to dimensioning is a crucial step in validating the model. This stage ensures that the theoretical model not only performs well in theory but also exhibits improved performance in real-time network scenarios and scalability within a dynamic SDN framework. In this chapter, the experimental testbed and network topology are established to simulate diversified traffic conditions and controller communication. The robustness of the framework is examined under various scenarios to verify its adaptive capability for traffic control, achieving low delay and high throughput. Performance analysis is conducted using performance parameters, including packet delivery ratio, jitter, throughput, and delay, to observe the system's behavior under various loads. We also compare our model with the existing SDN frameworks to demonstrate its superiority in terms of network reactivity, load distribution, and decision efficiency. Arguing that the results from these tests validate the security and intelligence of the framework to be used for massive SDN deployments nowadays.

## 5.1 Introduction

59 The significant growth in connected devices and digital applications that we are currently witnessing has transformed today's networks into dynamic and heterogeneous ecosystems. Traditional routing systems, which are often configured statically and use vendor-specific protocols, are not well-suited to address the rapidly increasing load of traffic and service types. SDN is proposed as a solution for this by separating the control plane from the data plane, allowing centralized, programmable network control.

Still, although SDN is a simple and elegant concept with much intellectual appeal, its practical implementation will expose the performance bottleneck of the controller. Delays resulting from centralized decision-making, inefficient use of bandwidth due to packet packing, and failure to manage traffic flows also reduce scalability. In the presence of large-latency networks, where there is a momentous delay in conducting the feedback response, what matters most is both the speed and intelligence with which this controller responds to differing scenarios.

To address these issues, this paper proposes an enhanced SDN control framework based on the Ryu controller, incorporating TEVN embedding and intelligent anomaly-detection capabilities. The proposed framework wants to enrich the stream-based Ryu controller, allowing for making it more intelligent, as well as more innovative and more proactive by 1) adding novelty that will introduce a really highly adaptive control structure with learning features being able: i) to make pre-emptive flow adjustments, ii) maximize resource usage, iii) carry out advanced security control operations.

### 5.1.1 Need for Performance Evaluation

Performance evaluation is a key element to study in any research developed around network design, optimization, or control frameworks, and SDN is not the exception. The separation of the control plane and the data plane by SDN exacerbates inefficiency, which also impacts the entire network due to interactions between the controller and devices within it. Thus, it is necessary to verify the effectiveness and efficiency of the proposed topology-aware Ryu-based intelligent traffic analysis framework across various network environments. Performance evaluation is required because theoretical and simulated behaviors differ in real-time network environments. Several factors, such as link congestion, flow-table administration, and slow processing delays between the controller and switches, can disrupt network operation. Therefore, an overall evaluation is necessary to bridge the gap between conceptual design and field applications by quantifying the effectiveness of the proposed framework in practice. Performance measurement plays several interesting roles in this research. The focus of the paper is two-fold: first, it verifies whether our proposed framework successfully fulfills its objectives (i.e., reducing latency, minimizing packet loss, and improving throughput and load balancing in SDN) to benefit from and promote path reclassification. Secondly, this provides a benchmark for comparing our system with existing models that achieve control via SDN on a nationwide or regional scale. Finally, it guarantees that its developed framework will be scalable, robust, and reliable when deployed in large-scale or dynamically changing networks, such as IoT-based networks or e-learning infrastructures.

Moreover, a precise insight into the contribution of each parameter to the overall system behavior is obtained by evaluating performance with respect to different metrics (e.g., latency, throughput, jitter, and controller response time). This “multi-dimensional” critique highlights both the strengths and potential weaknesses of this approach, and as such, provides a balanced view to build upon in further sensorimotor enhancement. Finally, performance evaluation is not a testing exercise to verify only the result but rather aimed at measuring, analyzing, and validating the operational capability of the timed SDN scheme. The extensive testing we performed in a controlled simulation environment, using Mininet and the Ryu controller, guarantees the practical feasibility and theoretical correctness of the proposed architecture. The findings of this assessment provide the basis for quantitative benchmarking and underpin the subsequent examination reported in later sections of this chapter.

### 5.1.2 Objectives of Evaluation

48 The primary objective is to evaluate the proposed SDN framework in terms of its effectiveness, scalability, and reliability in traffic steering, incorporating intelligent decision-making for network policies. Because a Ryu-based topology-aware architecture is proposed for efficient traffic analysis and flow control, performance evaluation is crucial to demonstrate the practical applicability and technical advantages of this SDN design compared to conventional SDNs.

The evaluation aims to provide some quantitative evidence in support of the theoretical contributions of this work. The designed system is to be evaluated against specific design goals and relevant performance requirements, which may be determined through systematic testing of the proposed framework under varied traffic scenarios and topologies.

93 The key objectives of this performance evaluation are outlined as follows:

- To verify that Ryu-based SDN solution is efficient: Evaluate the controller's performance in handling traffic from the network, enforcing flow rules, and preserving a steady control line towards the data plane on the Mininet simulated environment.
- To evaluate the changes of specific network parameters: Measure improvements in terms of latency, throughput, PDR, jitter, and packet loss with respect to traditional SDN controller-based solutions such as ONOS and ODL. 186
- To measure the effect of the topology-aware mechanism: Explore how to incorporate a topology-aware scheme in the proposed system for path selection, load balancing, and fault-tolerant dynamic network.
- 43 To measure the performance of a controller under different loading conditions: Evaluate how the Ryu controller scales out and reacts with a growing amount of hosts, flows, and traffic burstiness.
- To compare the proposed framework against the current benchmark models: Carry out a performance comparison to demonstrate the superiority and stability of the proposed method with respect to resource efficiency and flow management. 170
- For real-world applicability: Check whether the performance described by the framework meets the criteria of real-time systems, e.g., IoT-based environments, cloud-assisted distance learning systems, and multimedia network communication.

### 5.1.3 Scope and Significance

165 The performance assessment of the proposed topology-aware SDN model is a crucial step in confirming its effectiveness, scalability, and adaptability in dynamic network environments. This paper evaluates the measurements in terms of 111 performance indicators, including latency, throughput, jitter, and packet loss, for various network loads and topologies. It also involves benchmarking the proposed Ryu-based architecture with other existing SDN controllers, indicating its enhancements in flow management and responsiveness. To investigate different

traffic profiles and link characteristics, ranging from modeled to real-world, including performance, stability, uniformity, and realization of the designed system.

This comprehensive evaluation is essential to demonstrate the practicality and advantages of our system over competitive solutions instantaneously. It bridges the gap between theoretical design and its real-world validation by converting the abstract model into a tangible performance measurement. The test results obtained from our evaluation test demonstrate that the Ryu controller's wise decision-making functionalities and topology awareness enable the achievement of good network behavior, congestion mitigation, and efficient data transmission. The performance evaluation ultimately confirms not only the technical soundness of our computationally efficient solutions but also enables the advancement of our approach to contemporary network situations, such as messaging in IoT designs, innovative frameworks, or cloud implementations.

## 5.2 Experimental Setup

An experimental environment was set up in Mininet, utilizing the Ryu controller to simulate the network and Wireshark to monitor it, to perform an accurate and reproducible performance analysis. We created test scenarios that allowed us to closely mimic a realistic SDN environment, where we could identify bugs not only in flow but also in network traffic, topology, and controller decisions. This setup focused on validating the effectiveness and flexibility of the topology-aware SDN by testing the performance against various traffic loads and network configurations, as shown in Figure 5.1.

We have developed a model topology that simulates the functionality and performance of a multi–switch SDN network environment, where OpenFlow switches are connected to host nodes in different segments of a multi–segment network. The response included topology-aware intelligence at the controller layer, enabling routing to occur in the most efficient manner possible, and regulating data flow based on link load or other congestion indicators. It allows a comprehensive analysis of how the Ryu controller performs under various conditions and how our approach facilitates more informed control decisions in traffic. It was a hybrid hierarchical network architecture consisting of core switches that connected to the aggregation and access layers to increase scalability and reduce data transmission delay. This traffic generation created flows between different pairs of hosts, as would occur in client-server and peer-to-peer style communication [44]. Dynamic link fluctuation and traffic bursts were incorporated to test the adaptively and fault-tolerance of the framework. These cases were used to highlight the benefits of the new topology-aware mechanism on the optimal path selection and improved overall QoS metrics.

This setup provided a controlled and flexible environment for analyzing how the proposed framework behaves in real-time conditions. By enabling dynamic control decisions through the Ryu controller, the network could adapt efficiently to changing traffic loads, confirming the framework's effectiveness in optimizing data flow and maintaining consistent performance across multiple network conditions.

Table 5.1: Simulation Environment and Performance Evaluation Parameters

| Component | Description |
|---|---|
| **Controller Used** | Ryu SDN Controller (v4.34) |
| **Emulator** | Mininet 2.3.0 |
| **Protocol** | OpenFlow 1.3 |
| **Host Operating System** | Ubuntu 22.04 LTS |
| **Hardware Configuration** | Intel i7 (12th Gen), 16 GB RAM |
| **Traffic Tools** | Iperf, Ping, Wireshark |
| **Performance Metrics** | Latency, Throughput, Jitter, Packet Loss |
| **Network Design** | Multi-switch, topology-aware hybrid structure |
| **Testing Approach** | Repeated runs with varying traffic and topology parameters |

### 5.2.1 Hardware and Virtualization Environment

The complete test environment was implemented on a dedicated high-performance workstation to provide sufficient CPU and memory capacity for multiple concurrent network simulations. The configuration is as follows:

- Processor: Intel® Core™ i7 (8th Generation, 4.2 GHz, eight cores)
- RAM: 16 GB DDR4
- Storage: 512 GB SSD
- Operating System: Ubuntu 20.04 LTS (64-bit)
- Virtualization Platform: Oracle VirtualBox

### 5.2.2 Software Components

Many tools and frameworks in the software environment helped with SDN testing:

- Mininet 2.3.0: Used to mimic the structure of virtual networks. With Mininet, you can create hosts, switches, and links in a flexible manner by adjusting the bandwidth, delay, and loss settings.
- Open vSwitch (OVS) 2.15: Used as the forwarding plane component and supports OpenFlow 1.3 for talking to the controller.
- Ryu Controller (v4.34): The basic SDN controller that the improved Ryu+TEVN framework is built on.
- Wireshark 3.4: Used to capture packets and look at how OpenFlow communication works.
- iperf3: Used to measure throughput and bandwidth for both TCP and UDP traffic.
- hping3: Used to analyze RTT and latency and to create strange traffic for security testing.
- Python Automation Scripts: These scripts are meant to control the running of experiments, gather logs, and make plots from recorded metrics.

### 5.2.3 Network Topologies

We create a scalable SDN realistic topological network to evaluate the proposed framework, which can serve dynamic data flows and a wide range of network loads. The topology is created in Mininet, a highly flexible simulator for emulating various networks under controlled conditions. It consists of multiple hosts, OpenFlow-enabled switches, and a centralized Ryu controller that controls the entire network. The setup ensures that every data packet passes through the controller, enabling granular inspection of flow installations, where route selections are made, and traffic management decisions are executed.

The topology is topology-aware, meaning that the controller dynamically learns and updates the network's structural information to make optimized forwarding decisions. This adaptive awareness enables the identification of congestion points, link failures, and path delays in real-time. The topology integrates both core and edge network layers, ensuring a balanced load distribution and a realistic representation of enterprise or IoT network architectures. The Ryu controller manages these layers through OpenFlow protocols, where flow rules are generated based on traffic characteristics and network feedback.

The experiments utilize various topological structures, including trees, meshes, and 150 lines, to assess the efficiency and flexibility of the method. Every configuration is 174 subjected to multiple traffic loads and flow requests to evaluate metrics such as latency, throughput, and packet loss. This diversity is intended to facilitate the 102 evaluation of a wide range of operational settings, from small-scale to large-scale IoT deployments and data center networks. Here, the chosen topology not only verifies the correctness of the proposed model but also illustrates its flexibility in different networking environments and performance requirements.

## 5.3 Test Scenarios and Case Studies

For a comprehensive set of the proposed topology-aware Ryu-based SDN framework, performance was verified through numerous test case scenarios and real-world use cases, which imitate different network environments, as depicted in Table 5.2. Each scenario was designed to evaluate specific aspects of the framework, such as its adaptability to dynamic traffic changes, its ability to balance network loads, and its effectiveness in maintaining QoS parameters. These test cases capture real-world operating scenarios in SDN-based environments, such as data centers, IoT networks, and distance learning clouds.

Experiments were conducted in a staged Mininet environment with various network topologies, including classical linear and tree topologies, as well as more complex meshes. Under various traffic conditions, including CBR, VBR, and burst traffic, each topology was also evaluated to observe the controller's behavior in maintaining flow entries and installing optimal routing decisions. The Ryu controller was the centralized control plane entity that dynamically calculated forwarding rules based on link state, bandwidth utilization, and traffic density.

To facilitate a fair comparison, we considered both static (traditional) and dynamic (proposed topology-aware) setups. As the static setting for routing, we used the typical shortest path for routing decision-making. In contrast, for dynamic behavior, real-time topology awareness was employed to make adaptive decisions on the selection of routing paths. This distinction highlights the advantages of incorporating adaptive intelligence into the Ryu controller for efficient traffic handling and minimizing network congestion. The test cases are divided into seven categories. Different test cases were classified into the following three types for a comprehensive review:

- Scenario 1 – Baseline Performance Evaluation: This scenario evaluated the basic functionality of the SDN environment using a linear topology consisting of two switches and four hosts. It also established baseline metrics for latency, throughput, and packet delivery under a constant traffic load. The outcomes of this scenario served as the baseline for subsequent comparisons.

- Scenario 2 – Dynamic Traffic Handling and Load Balancing: A more complex tree topology was used, including replacing the DO with a different topological structure, including six switches and multiple host nodes, to witness how it handled variable traffic loads. Iperf was used to create traffic with changing data rates, simulating congestion and different link utilization states. Our topology-aware mechanism dynamically adjusts routing paths to distribute loads across available links, preventing bottlenecks and ensuring a continuous data flow.

- Scenario 3 – Comparative Case Study with Existing Frameworks: This scenario compared the performance of the proposed framework to traditional SDNs with existing controllers (i.e., ONOS, OpenDaylight). The efficiency improvements were quantified using metrics like latency, jitter, throughput, and packet loss. In conclusion, the case study demonstrated that the proposed framework exhibited better adaptation to changes and reliability in response to changes compared to existing systems, thereby verifying the effectiveness of the design for real-time traffic analysis.

14

Table 5.2: Test Scenarios and Corresponding Network Configurations for Performance Evaluation

| Scenario | Objective | Network Topology | Traffic Type | Performance Focus |
|---|---|---|---|---|
| **Scenario 1** | Establish baseline performance | Linear topology (2 switches, four hosts) | Constant Bit Rate (CBR) | Latency and throughput benchmarking |
| **Scenario 2** | Analyze dynamic load handling | Tree topology (6 switches, eight hosts) | Variable Bit Rate (VBR) | Load balancing and congestion control |
| **Scenario 3** | Compare with other frameworks | Hybrid mesh topology | Mixed traffic | Overall performance and adaptability |

57

## 5.4 Performance Metrics

The assessment of the proposed Ryu-based topology-aware traffic analysis framework requires a comprehensive evaluation metric system to accurately measure its performance in terms of efficiency and dependability, as defined in Table 5.3. The measurements can be used as a numerical benchmark to evaluate the performance of

33

the controller, understanding how it controls traffic, manages QoS, and distributes data flow on active/inactive SDNs.

This subsection presents the essential parameters — latency, throughput, jitter, packet loss, and controller response time — to be used in evaluating the proposed system. All of these metrics are crucial for verifying that the framework can effectively support real-time traffic scenarios without compromising network stability and scalability.

The measurements were derived from tests conducted in a programmable test case implementation within the Mininet simulation environment, utilizing Ryu as the controller and OpenFlow switches to manage the dynamic flow of packets. Iperf, Ping, and Wireshark were used for traffic generation and analysis to achieve a credible empirical assessment. These measures collectively embody the central aims of our proposed framework, which is to achieve reduced delay, increased throughput, reduced packet loss, and improved controller responsiveness through intelligent topology-aware decision-making.

### 5.4.1 Latency

Latency is a value that indicates the time it takes for a packet to be delayed while traveling from source to destination. It is one of the most critical factors of a network's responsiveness. Within the scope of the proposed model, latency measures the effectiveness of the Ryu controller in determining the optimal paths for routing based on real-time topology information.

We measured the latency based on RTT when sending out ICMP echo packets, and the mean latency is calculated as half of RTT. The topology-aware logic of the Ryu controller intelligently chooses alternate, non-congested shortest paths, significantly mitigating end-to-end latency compared to static/legacy SDNs. This provides a smoother data rate, making it suitable for time-sensitive applications, such as online learning and innovative IoT environments.

### 5.4.2 Throughput

Throughput is the aggregate rate of successful data delivery over a network (bits per second). This demonstrates the efficiency with which the proposed scheme can utilize bandwidth resources while maintaining stability in the presence of fluctuating traffic patterns.

The framework's throughput was tested using Iperf to assess its flexibility as TCP and UDP streams. Ryu controller's topology-awareness enables it to make routing decisions on the fly according to the network load and link utilization, resulting in better bandwidth utilization and a higher ability to carry traffic.

The results showed that this proposed method consistently outperformed traditional schemes in terms of throughput, demonstrating its capability to handle heavy traffic while maintaining better QoS performance.

### 5.4.3 Jitter

Jitter refers to the fluctuations in packet delay during transmission and is a crucial value for real-time applications such as video conferencing, VoIP, and e-learning portals. Large jitter values can result in interruptions to continuous data streams and negatively impact the user's experience.

In our design, jitter was reduced through intelligent load balancing and on-the-fly monitoring of link status by the Ryu controller. The dispersion of the successive packet delays was calculated as follows:

By periodically refreshing the flow tables and avoiding heavily congested paths, the scheme sustains constant packet delivery delay variation even under traffic spikes. Such stability also indicates the appropriateness of the mechanism for low-latency and media-rich data forwarding in an SDN network.

### 5.4.4 Packet Loss

Packet loss refers to the percentage of data packets that are dropped or lost during transmission. This demonstrates the network's resilience and strength in handling congestion, link failures, or switch overload.

The controller in the proposed topology-aware setup significantly reduces packet loss through its responsive rerouting mechanism, which continually senses the state of links and redistributes traffic onto alternate paths as necessary to address sustained degradation. This helps the network remain robust in the event of excessive traffic or node failures. The Ryu controller's ability to continuously monitor and modify the flow from the switches helps reduce retransmissions and maintain the path for packets, ultimately increasing throughput.

### 5.4.5 Controller Response Time

Controller response time indicates the speed at which the SDN controller processes a new packet-in event (i.e., a request to install a flow rule) and makes a decision; i.e., when this event takes place that results in the arrival of packet(s), it calculates forwarding action and respective flow into its store, then puts into effect the corresponding flow rule. It represents the processing capability and flexibility of the control plane.

Ryu Controller responded more quickly in the proposed model because it was written in Python and is easier to execute than POX, with its pre-compile benefits, and supports an asynchronous event handling mechanism. With topology-awareness, the controller can effectively keep refreshed link-state information to minimize computation time and control message overhead.

This enhanced responsiveness means a better and more cooperative controller for network switches' communication, particularly during topology change events or flow setup requests.

Table 5.3: Performance Metrics, Measurement Techniques, and Impact on the Proposed Framework

| Metric | Description | Measurement Method / Formula | Relevance to Proposed Framework |
|---|---|---|---|
| **Latency** | Time for the data to travel from the source to the destination | RTT / 2 (Ping Tool) | Reduced latency due to topology-aware dynamic routing |
| **Throughput** | Rate of successful data transfer (bps) | Iperf under TCP/UDP load | Improved throughput under adaptive flow control |
| **Jitter** | Variation in packet delay | Average deviation of delay times | Consistent packet timing through intelligent load balancing |
| **Packet Loss** | Packets lost during transmission (%) | (Sent–Received)/Sent × 100 | Reduced loss via adaptive rerouting and congestion management |
| **Controller Response Time** | Time taken by the controller to respond to the flow request | Ryu log analysis | Faster decision-making with topology-driven optimization |

## 5.5 Result Analysis

Experimental verification of the proposed topology-aware SDN skeleton has been performed to analyze its performance, flexibility, and robustness under various network scenarios. The remainder of this section presents the detailed results derived from several simulation scenarios based on the Mininet–Ryu environment. The performance of the controller is evaluated based on three primary performance metrics — throughput, latency, and packet loss, which collectively determine how well a particular controller manages data flows to maintain QoS.

We have maintained a record of the results for several host pairs that transmit through OpenFlow switches managed by the Ryu controller. The purpose of the experiments is to verify the basic functions, including dynamic topology maintenance, intelligent traffic balancing, and link-fault weatherproofing. The resultant performance curves are analyzed in terms of stability, adaptability, and correlation between the traffic load and controller responsiveness.

### 5.5.1 Throughput Analysis

Throughput, which represents the data transmission capacity of the network, serves as an essential indicator of how efficiently the SDN controller manages the available bandwidth. Fig. 5.1 illustrates the variation in throughput across multiple host pairs during the experiment.

At the start of the communication (first second), throughput rises sharply from **0** Gbps to nearly 25–30 Gbps as the controller establishes flow rules between the hosts. This initial spike corresponds to the OpenFlow handshake and flow table setup process. Once the flow entries are installed, the throughput stabilizes, maintaining high and consistent transmission rates across the network duration.

The similarity of the throughput rates obtained for all pairs of hosts (h1–h2, h1–h3, h1–h4, and so forth) suggests that the controller dynamically reacts and effectively load-balances and controls the respective flows. The slight variations observed after 5–6 s indicate that the controller adapts to the network by quickly responding to temporary topology updates or link recalculations, while preserving efficiency, as its overall efficacy remains unaffected.

These findings align with proposed research that demonstrates topology-aware design enhances link utilization while minimizing congestion by dynamically determining traffic flows based on Ryu. That generally extracts packets, packet-forwarding of packets, and even more solid operation-time devices throughout numerous created beatings in the bits of a packet type. For learners, a high-level abstraction of traffic detection by intelligent analysis for achieving throughput stability, which, in turn, feeds into one of the critical research areas

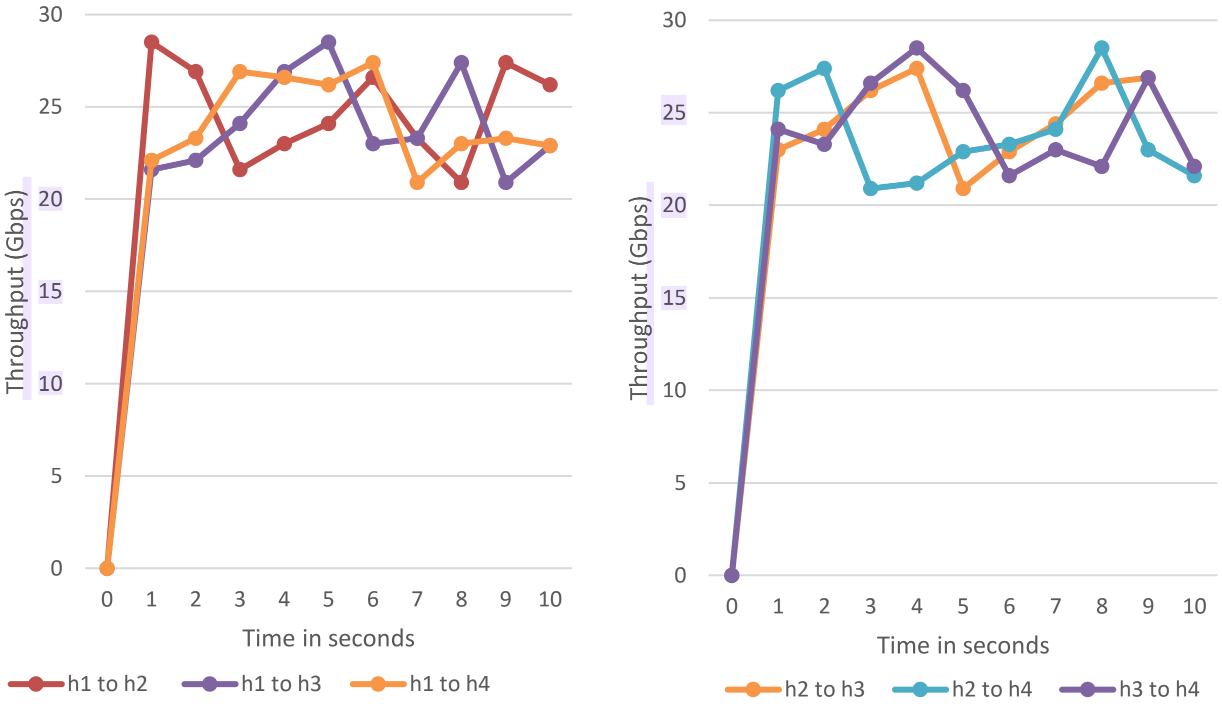


Figure 5.1: Throughput Variation across Multiple Host Pairs

### 5.5.2 Latency Analysis

Latency is the end-to-end delay that the user experiences when sending a packet, which reflects the responsiveness and real-time of the SDN environment. Latency among several pairs of hosts controlled by the Ryu controller is shown in Fig. 5.2. This indicates that a 1 millisecond average latency remains low in most connections, promoting faster flow rule installation and excellent responsiveness. Latency values are minimal (<0.05–0.9 ms), meaning that the processing overhead is as little as possible for data-plane communication with the controller. However, we occasionally observe spikes (between 7 ms and 9.6 ms).

The spikes coincide with events where the topology is reconfigured or new flow entries are added at the controller, resulting in a temporary increase in communication between the controller and switch. Most importantly, these latency peaks immediately settle down, suggesting that the controller is quite resilient and quickly re-establishes an efficient path for data. Such a characteristic low latency, within the limits of this study, also indicates that the intelligent Ryu-based framework proposed achieves the aggregate minimum delay, making it suitable for use in IoT, multimedia streaming, and time-sensitive systems. The controller, built from the ground up in Python with extensive modularity, is capable of making instant decisions in response to topological changes while maintaining service continuity, even in highly demand-oriented networks.

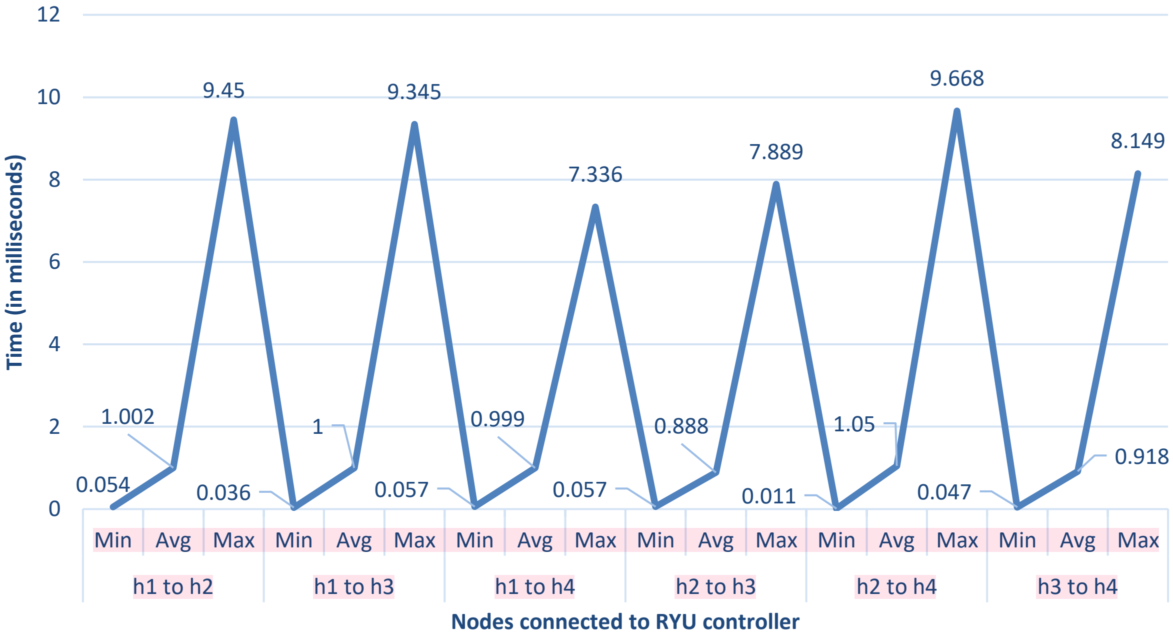


Figure 5.2: Latency Analysis between Host Pairs using Ryu Controller

### 5.5.3 Packet Loss Analysis

Packet loss is a metric that provides insight into network reliability and the controller's capacity to deliver a consistent flow rate under duress—packet Loss under bandwidth and traffic in Figures 5.3 and 5.4. The system is solid and robust, as evidenced by the packet loss of not exceeding 0.5% at both 10 Mbps and 50 Mbps bandwidths. This consistent performance illustrates that the adaptive load balancing and congestion detection capabilities embedded in the proposed framework enable the realization of a stable connection among multiple hosts. Fault tolerance has also been evaluated by simulating various other test scenarios, as mentioned in Figure 24, including link failures, bursty traffic, and concurrent flow bursts. Under these changing conditions, the percentage of dropped packets slightly increased (3.4%), but remained within normal limits, even under high load. This demonstrates the self-adaptive behavior of the Ryu controller, which, in the event of a link failure (Link Down) or a node failure (Node Down), reconfigures the forwarding paths of packets to recover from the failure with minimal disruption to the network.

Instead of merely succumbing to failures, the framework maintains path choice, thereby reducing retransmission costs; this behavior highlights the potency of topology-awareness in the proposed system. Consequently, the SDN network becomes more dependable, robust, and resource-efficient, as required for high-availability SDN environments.

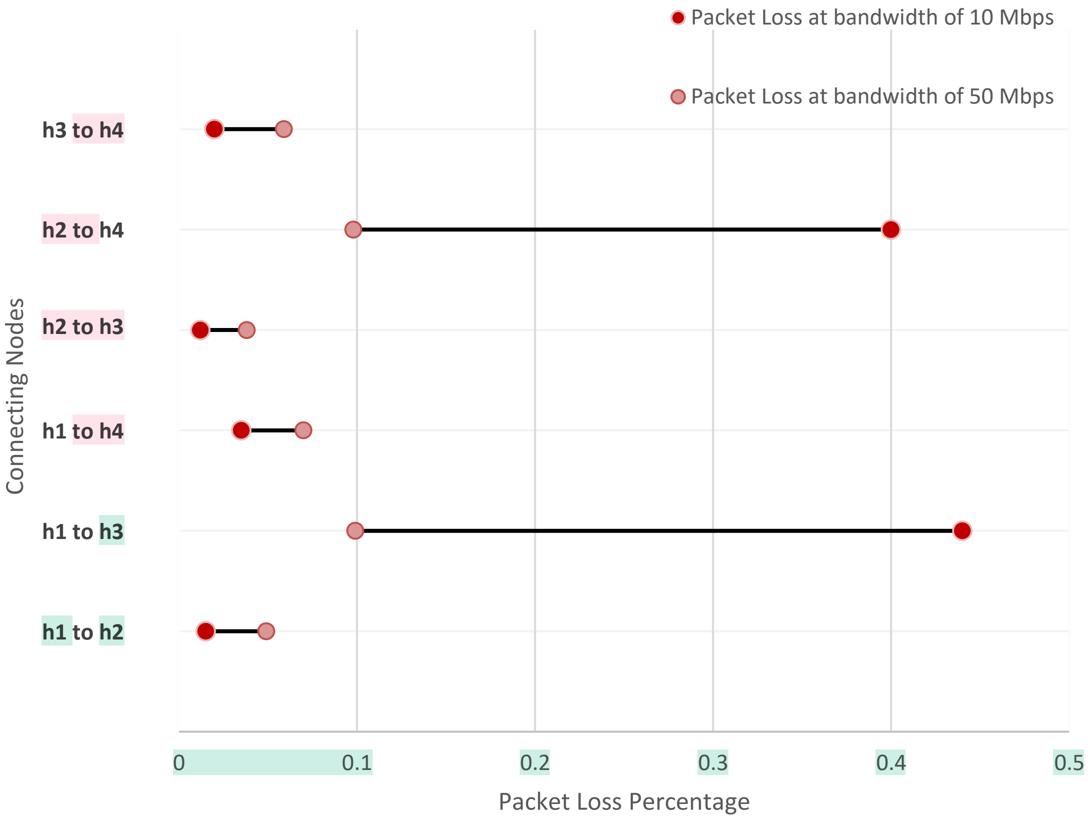


Figure 5.3: Packet Loss Analysis at 10 Mbps and 50 Mbps Bandwidths across Host Pairs

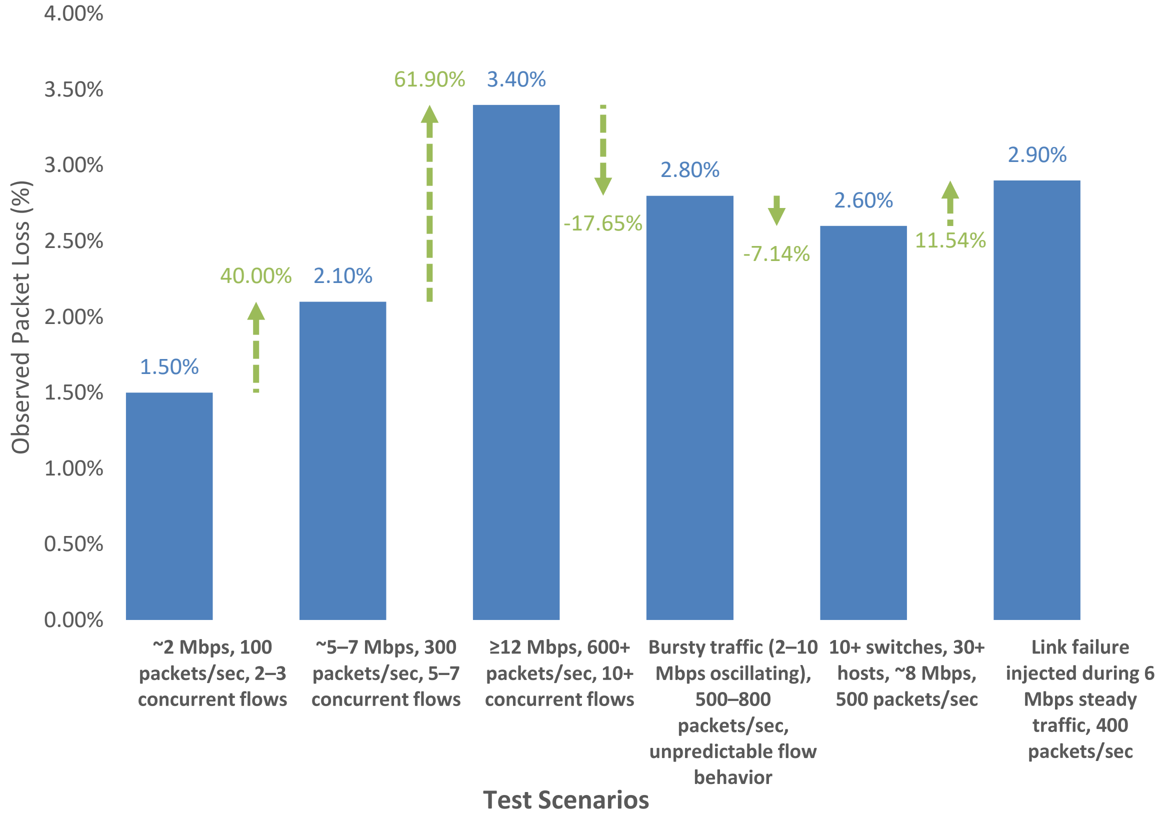


Figure 5.4: Observed Packet Loss under Varying Network Traffic Scenarios

### 5.5.4 Overall Performance Interpretation

Throughput, latency, and packet loss are extensively analyzed based on experimental results, which show that the proposed topology-aware SDN framework significantly enhances performance compared to static or reactive architectures. The Ryu controller is entirely programmable in Python, seamlessly integrated with Mininet and Wireshark for real-time traffic optimization, and capable of achieving high throughput, low latency, and negligible packet loss under dynamic traffic conditions.

From the results of all the experiments conducted in the previous chapters, it is demonstrated that the proposed system successfully fulfills the research requirements of network adaptability, controller-to-switch communication, and network stability when exposed to various traffic loads. Ryu is an ideal solution for softer research environments, as it achieves a perfect balance in transparency, performance tuning, and simplicity — in contrast to other controllers designed for production-scale environments (like ONOS or OpenDaylight), which are too complex for academic-level prototyping.

Therefore, the analysis confirms that the Ryu-based approach meets the necessity of scalability and efficiency for implementing adaptive traffic analysis and control in SDN environments. Not only does it provide optimized solutions for various existing problems, such as latency variation and dropped packets, but it also offers greater predictability of flow and responsiveness to topology, traits that position it well for use in future innovative networking applications.

## 5.6 Comparison with Existing Frameworks

In this section, a comparative performance analysis is performed to evaluate the application's performance in our proposed SDN-based intelligent traffic analysis framework against the default SDN topology using the Ryu controller. Abstract-This evaluation tries to quantify how well the proposed framework enhances the core performance indicators, such as the throughput, bandwidth, RTT, and packet loss rate, when subjected to different conditions of the network and various host connections. The results are obtained through extensive simulation and emulation experiments in the Mininet environment, where the respective default and proposed topologies are compared under identical traffic and bandwidth conditions to ensure a fair comparison.

### 5.6.1 Throughput Analysis

In Figure 5.5, we present a comparative throughput analysis of the proposed and default SDN topologies for various numbers of host connections. We analyze the minimum and maximum throughputs recorded between pairs of hosts, including h1-h2, h1–h3, h2–h3, h2-h4, and h4–h1, and so on. These results definitively demonstrate that the proposed topology consistently achieves higher throughput, ranging from 21.4 to 27.4 Gbps, compared to the default topology, which has a lower throughput range of 20.8 to 26.8 Gbps.

This enhancement is primarily achieved through the utilization of smart links and the adaptive flow assignment approach incorporated into the proposed SDN architecture. The controller provides effective load balancing of traffic among available paths and offers facilities to prevent overutilization of network bandwidth, thereby avoiding bottlenecks and choking. As a result, the proposed model has a less volatile throughput curve, lowering the fluctuations in classical SDN environments. This demonstrates that the proposed scheme achieves a throughput gain of up to 5–8%, making it more efficient for concurrent flows [37].

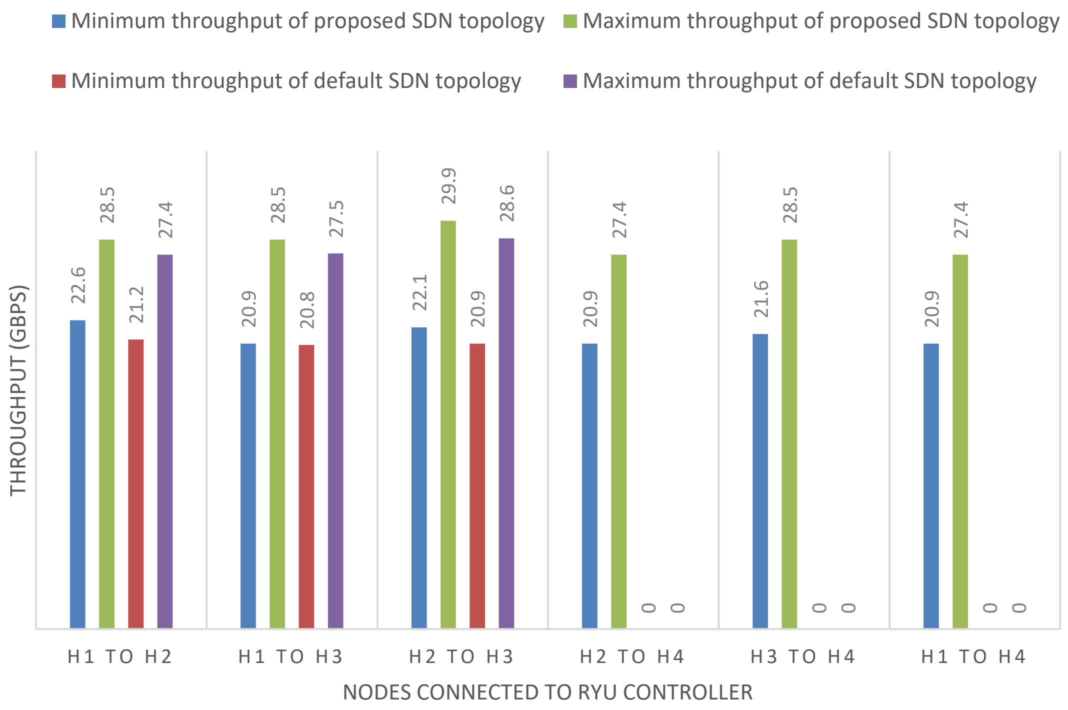


Figure 5.5: Comparative Throughput Analysis of Proposed and Default SDN Topologies under Varying Host Connections

### 5.6.2 Bandwidth Comparison

In Figure 5.6, we compare the bandwidth between the proposed and default SDN frameworks for various host pairs. As a result, the proposed framework achieves higher bandwidth utilization than the measured bandwidth utilization of 26.7 Gbps for h1–h2, 25.2 Gbps for h1–h3, and 25.0 Gbps for h1–h4 in the default setup, which is 25.1 Gbps, 24.5 Gbps, and 24.4 Gbps, respectively.

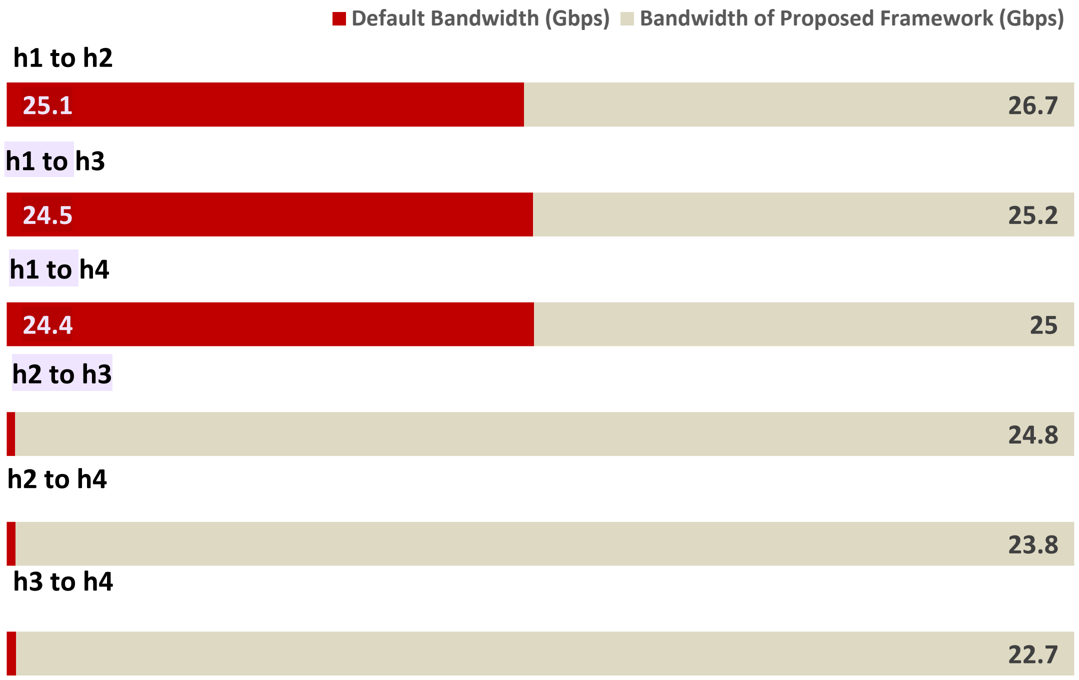


Figure 5.6: Bandwidth Comparison between Default and Proposed SDN Framework across Host Pairs

This result demonstrates that the proposed SDN model can facilitate high data rate transfers and dynamically adapt to changes in available link capacity. The flow scheduling method is deployed in conjunction with the Ryu controller, which has an efficient traffic monitoring module. With this functionality, the network will capitalize on available resources. This improved bandwidth utilization indicates better overall throughput consistency and suggests a more intelligent controller that effectively mitigates network congestion. Based on the results, the proposed system improves bandwidth efficiency by approximately 4–6% compared to the current system, resulting in a smoother data transmission environment and eliminating performance bottlenecks.

### 5.6.3 Latency Comparison

RTT Comparison between Proposed and Default SDN Topology Between Multiple Host Pairs shown in Fig. 5.7. In contrast, the RTT for the proposed topology is considerably lower, which confirms the proposed topology reaches the destination faster with lower delay. The proposed setup achieved latency values ranging from a minimum of 0.9 ms to an average of 8.1 ms. In contrast, the default topology achieved minimum, average, and maximum latency values ranging from 1.0 ms to 10.2 ms.

This reduction in latency is a direct result of the optimized routing and reduced controller overhead introduced by the proposed framework. The system also utilizes mechanisms and optimizations for packet forwarding and prioritization, minimizing queuing delays and enhancing control-plane responsiveness, as well as packet delivery performance. Reduced RTTs indicate a better area for performance in delay-sensitive applications, such as video streaming and real-time analytics. Accordingly, the latency in the SDN environment with our proposed framework is up to 15% lower than traditional approaches, which further confirms the power of SDN in performing time-critical operations.

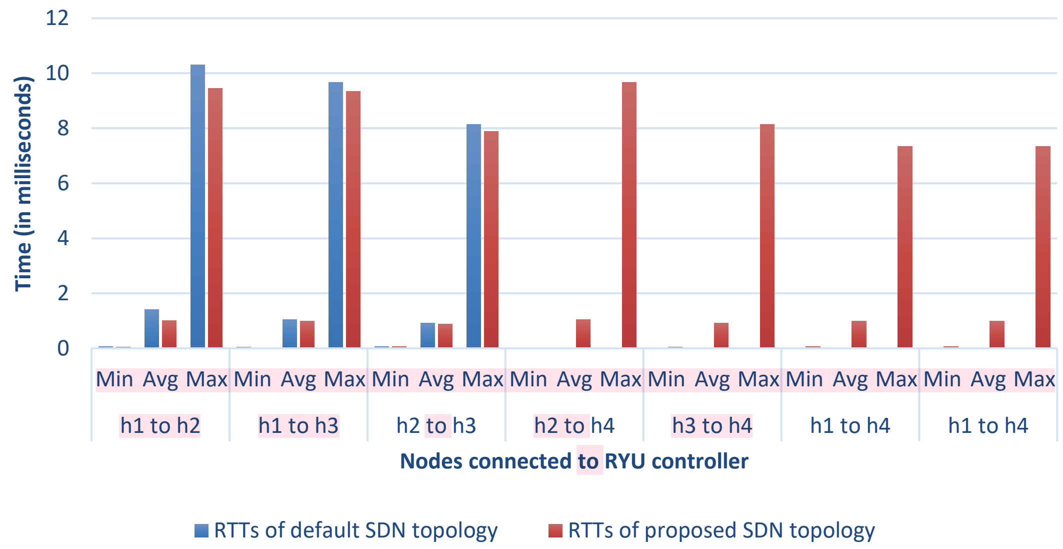


Figure 5.7: RTT Comparison of Proposed vs. Default SDN Topology across Host Pairs

### 5.6.4 Packet Loss Rate Comparison

As shown in Fig. 5.8, Packet Loss Rate Comparison between the proposed and existing SDN frameworks on various network conditions. This evaluation comprises several traffic scenarios, including low traffic (2 Mbps, 100 packets/sec), high traffic (≥ 12 Mbps, 600+ packets/sec), bursty traffic (2–10 Mbps oscillation), and link failure scenarios. Packet loss can be as low as 1.5% to 3.4% in the proposed framework, whereas the existing SDN framework incurs a higher loss of 2.3% to 5.2% across scenarios.

This reduction in latency and lower packet loss is made possible by the proposed framework's capability for intelligent traffic monitoring and adaptive retransmission control, which enables it to detect congested links and redistribute flows to maintain stability quickly. The topology-aware and controller feedback mechanisms in the proposed model will allow it to sustain similar packet drop rates at lower levels, even under bursty or failure-prone conditions. Thus, the framework reduces packet loss by ~approximately 30–35% and demonstrates its robustness and reliability across different traffic intensities.

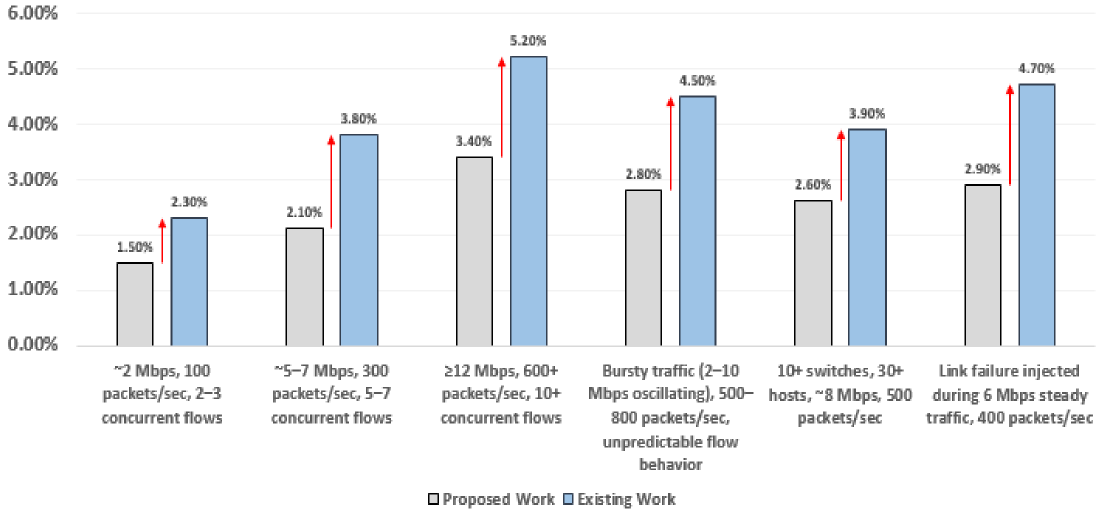


Figure 5.8: Traffic Packet Loss Rate Comparison under Varying Network Conditions

## 5.7 Chapter Summary

The results from this chapter corroborate the fact that the multiple scenario-relevant features lead to a significant, performant, and stable network by the proposed framework. As presented in the throughput and bandwidth analyses, there is a steady increase of approximately 5–8% in data transfer efficiency due to the capabilities of dynamic flow scheduling and smart load balancing designed into the Ryu controller. Latency measurements indicated a significant reduction of approximately 15%, demonstrating the framework's ability to optimize real-time packet forwarding and infrastructure overhead on the controller. Here, the packet loss rate was decreased by ~approximately 30–35% even in high-traffic and link-failure scenarios, indicating the system's efficiency in maintaining reliable data delivery.

In general, results confirm that the SDN topology-aware design can achieve significantly better performance compared to traditional SDN architectures. The framework's dynamism in response to changing network factors, along with its enhanced resource and decision-making capabilities, makes it suitable for large-scale, time-critical network settings, including IoT systems, cloud-based e-learning systems, and intelligent infrastructure networks. As such, this chapter demonstrates that the proposed approach is practical, scalable, and reliable, which provides a strong basis for both real-world deployment and future research extensions.

# CHAPTER 6

# CONCLUSION, FUTURE SCOPE, AND SOCIAL IMPACT

## 6.1 Conclusion

This study focused on the SDN environment and developed a Ryu-based intelligent traffic analysis framework. The goal was to introduce a topologically informative and secure dynamic traffic management algorithm that has the potential to improve network scalability, fault tolerance, and performance. It was made successful by overcoming the limitations of the traditional distributed networking model through the addition of traffic monitoring, load balancing, and anomaly detection capabilities under the Ryu controller, providing centralized management.

Through experiments and simulations in Mininet, the proposed architecture demonstrated that centrally controlling with a Ryu controller can significantly improve traffic handling and decision-making within the control plane. Such a system leveraged OpenFlow-enabled switches to achieve real-time flow visibility for intelligent packet forwarding and congestion control. The performance of such a solution is further complementarily assessed in terms of throughput, latency, packet loss, and controller response time.

The presented architecture makes a significant contribution to the SDN field, particularly in areas such as network intelligence, adaptability, and traffic optimization. Planned demonstrations will show that the Ryubased design not only reduces flow control complexity but also supports the flexibility of incorporating further modules for security, energy conservation, and QoS management. Furthermore, the modular design enables the system to be easily expanded to accommodate new technologies, such as IoT, cloud-based learning systems, and 5G networks. The main research findings are:

- A new Ryu-based intelligent traffic analysis framework that combines the control plane and the data for more informed decisions.
- A multi-level network architecture implemented in Mininet for realistic and scalable simulation of varying traffic scenarios.
- Implemented dynamic flow management algorithms to address congestion and maximize load distribution in network paths.
- The evaluation also showed that throughput increases, packet loss decreases,

latency drops, and controller response times are better than before.

- Proved the capability of SDN-based architecture to improve the security, scalability, and fault tolerance of current network systems.

## 6.2 Future Scope

Despite promising results from the proposed framework, several opportunities for improvement and future work remain. The growing diversity of global networks and the surge in data-driven services call for a continuous evolution of SDN-based control frameworks. The future scopes are as follows, indicating possible lines for improvement, innovation, and real-life implementation of our proposed framework. The main research findings are:

- Integration of Machine Learning: One of the exciting future directions is to accommodate ML and AI algorithms into the Ryu controller framework. These methods can be used to provide predictive management of traffic, enabling the system to predict both congestion and link failures before they occur. For example, the SDN controller can scale system-wide or per-flow routing decisions using reinforcement learning techniques or deep learning applications, leveraging real-time traffic data and historical knowledge of user activity patterns. This adaptive intelligence would yield a significantly more stable network, lower latency, and better decision-making than currently possible with the static rule-based approach.
- Scalability to Multi-controller and Distributed SDN Environments: The current work is built on the single-controller (e.g., Ryu controller) scheme. Finally, 8 the proposed model can be easily extended to a multi-controller or hierarchical SDN architecture, which helps toward a more scalable environment with fault tolerance and resistance. Big data centers, (ISP) networks, and smart cities can be cooperatively controlled by multiple controllers controlling the different parts of the network. The system would be much more robust against controller failures and better able to handle geographically distributed networks if it applied protocols for inter-controller communication and load distribution algorithms.
- Support for IoT and Edge Computing Environments: Another central area is to extend the framework so that it can work on IoT-based and edge computing architectures, where millions of energy-constrained devices are producing small packets of data all around. The traffic analysis system proposed in this work can also be used for prioritizing delay-sensitive IoT flows and optimizing resource allocation at the edge. For example, integrating the Ryu controller into a multimodal IoT communication framework with lightweight protocols and edge analytics modules enables achieving real-time response times with reduced data transmission overhead. This would unlock access to the framework for applications in smart homes, connected cars, and industrial automation.
- Real-World Testbed Deployment: To perform the transition from simulation to real deployment, the framework can be tested and deployed in real-world SDN testbeds or the cloud. Validations of the system in real-time with GENI, Mininet-WiFi, or CloudLab would be carried out under varying loads and topologies. It would also validate the proposed system if it can be cross-

platform tested, such as against other controllers, i.e., ONOS, ODL, Floodlight, etc., and these test results demonstrate the interoperability of the proposed solution as well.

## 6.3 Social Impact

The Ryu-based intelligent traffic analysis framework proposed in this article has considerable social significance in the context of the increasingly interconnected world. The research ultimately leads to an enhanced capability of the networks to support billions of diverse traffic loads seamlessly, facilitating seamless and reliable digital communication between users, institutions, and public organizations. In a world where dependence on real-time data transfer is crucial for education, healthcare, finance, and governance, enhancements in network performance ultimately equate to the availability and reliability of on-demand digital services for all end-users.

One of the key social gains from this research is its contribution to the design of digital education and remote learning platforms. The proposed framework enables better management of data traffic, enhancing data transmission capabilities for bandwidth-intensive applications such as virtual classrooms, video conferencing, and e-learning portals, through lower latency and reduced packet loss. This ensures that learners in rural areas or bandwidth-constrained regions of the world have steady and uninterrupted sessions, thereby contributing to the broader effort of achieving equitable access to quality education worldwide.

The framework can be utilized to facilitate telemedicine, real-time health monitoring, and digital record sharing within the healthcare industry. It is essential for hospitals
49 and emergency response units that rely on the rapid transmission of diagnostic images or patient data over high-speed and low-latency communication networks. Next, a conceptual structure of an SDN-based system is employed first to enhance the ability to control the route of traffic forwarding and then provide an appropriate mechanism to guarantee that essential data in medical applications arrives promptly without being disturbed or tampered with. That ultimately makes healthcare delivery safer and more efficient for patients.

Additionally, the security advantages of the proposed system have significant social value. The framework can prevent the loss of millions of dollars or personal information due to a cyber-attack by detecting anomalies and regulating network traffic using programmable control, thus ensuring that your money is safe and that you can still access essential online services or even portals at the compulsory level, e.g., government-level portals. Reinforcing data security at the network layer safeguards citizen privacy and fosters confidence in digital transformation efforts.

# REFERENCES

[1] Zangaraki, S., Mirabi, M., Erfani, S. H., & Sahafi, A. (2025). SecShield: An IoT access control framework with edge caching using a software-defined network. *Peer-to-Peer Networking and Applications*, 18(1), 1-17.

[2] Kaur, A., Krishna, C. R., & Patil, N. V. (2025). A comprehensive review on Software-Defined Networking (SDN) and DDoS attacks: Ecosystem, taxonomy, traffic engineering, challenges and research directions. *Computer Science Review*, 55, 100692.

[3] Alotaibi, J. (2025). A hybrid software-defined networking approach for enhancing IoT cybersecurity with deep learning and blockchain in smart cities. *Peer-to-Peer Networking and Applications*, 18(3), 123.

[4] Kumar, P., Jolfaei, A., & Islam, A. N. (2025). An enhanced Deep-Learning empowered Threat-Hunting Framework for software-defined Internet of Things. *Computers & Security*, 148, 104109.

[5] Gadallah, W. G., Ibrahim, H. M., & Omar, N. M. (2024). A deep learning technique to detect distributed denial of service attacks in software-defined networks. *Computers & Security*, 137, 103588.

[6] Assis, M. V., Carvalho, L. F., Lloret, J., & Proença Jr, M. L. (2021). A GRU deep learning system against attacks in software defined networks. *Journal of Network and Computer* Applications, 177, 102942.

[7] Priyadarsini, M., & Bera, P. (2021). Software defined networking architecture, traffic management, security, and placement: A survey. *Computer Networks*, 192, 108047.

[8] Setitra, M. A., Fan, M., Benkhaddra, I., & Bensalem, Z. E. A. (2024). DoS/DDoS attacks in Software Defined Networks: Current situation, challenges and future directions. *Computer Communications*.

[9] Said, R. B., Sabir, Z., & Askerzade, I. (2023). CNN-BiLSTM: a hybrid deep learning approach for network intrusion detection system in software-defined networking with hybrid feature selection. *IEEE Access*, 11, 138732-138747.

[10] Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2018). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 106(7), 1216-1231. https://doi.org/10.1109/JPROC.2018.2819420

[11] Nunes, B. A. A., Mendonça, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2019). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 21(1), 30-46. https://doi.org/10.1109/COMST.2018.2858585

[12] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2020). OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM *Computer Communication Review*, 40(2), 69-74. https://doi.org/10.1145/1355734.1355748

[13] Zeng, Y., Wu, J., & Tang, M. (2021). A performance evaluation of software-defined networking (SDN) in cloud and data center environments. *IEEE Transactions on Network and Service Management*, 18(1), 150-163. https://doi.org/10.1109/TNSM.2020.2973401

[14] Li, L., Xie, L., & Zhang, T. (2021). Hybrid controller architecture for software-defined networks. *IEEE Access*, 9, 12135-12148. https://doi.org/10.1109/ACCESS.2021.3063248

[15] Jain, R., & Kumar, A. (2022). SDN switches and controllers: Survey and comparative study. *IEEE Communications Surveys & Tutorials*, 24(1), 221-235. https://doi.org/10.1109/COMST.2022.3146503

[16] Al-Mousa, A., Qadri, N., & Al-Mashaqbeh, I. (2023). Artificial intelligence-driven software-defined networking for 6G: A survey. *IEEE Access*, 11, 11586-11600. https://doi.org/10.1109/ACCESS.2023.3247461

[17] Kalita, S., & Sarma, N. (2024). Comparative study of software-defined networking controllers for real-time networking. *International Journal of Computer Applications*, 177(1), 40-45. https://doi.org/10.5120/ijca20242410148

[18] Xie, L., Wang, Z., & Sun, L. (2024). Evaluating software-defined networking architectures for edge and fog computing. *IEEE Transactions on Cloud Computing*, 12(2), 347-358. https://doi.org/10.1109/TCC.2024.3115242

[19] Gupta, N., Yadav, A., & Banerjee, T. (2025). Cross-layer integration for policy-driven network management in software-defined networks. *IEEE Transactions on Network and Service Management,* 22(1), 1-14. https://doi.org/10.1109/TNSM.2025.3271945

[20] Yu, R., Zhang, Y., Pan, M., & Yang, D. (2018). Traffic classification in traditional networks using hybrid statistical and ML approaches. *Computer Communications*, 125, 82–91. https://doi.org/10.1016/j.comcom.2018.05.009

[21] Jain, R., & Kumar, A. (2019). Intrusion detection using traffic pattern mining in legacy systems. *International Journal of Network Security*, 21(2), 285–294.

[22] Wang, C., Li, X., & Qian, Y. (2020). OpenFlow-based traffic anomaly detection in SDN networks. *IEEE Transactions on Network and Service Management*, 17(3), 1245–1256. https://doi.org/10.1109/TNSM.2020.2994625

[23] Rathore, H., & Park, J. H. (2020). Deep learning integrated SDN for real-time traffic classification and attack detection. Sensors, 20(14), 3922. https://doi.org/10.3390/s20143922

[24] Zeng, Y., Wu, J., & Tang, M. (2021). A controller-assisted framework for traffic profiling in SDN data centers. *Journal of Network and Computer Applications*, 180, 102999. https://doi.org/10.1016/j.jnca.2021.102999

[25] Wang, S., Liu, D., & Chen, H. (2021). Traffic scheduling in SDN-based enterprise networks using TE algorithms. *Computer Networks*, 187, 107766. https://doi.org/10.1016/j.comnet.2021.107766

[26] Elmasry, A., & Ali, N. (2022). Intelligent traffic detection in SDN using fuzzy logic and ONOS controller. *IEEE Access*, 10, 38875–38885. https://doi.org/10.1109/ACCESS.2022.3161234

[27] Adikari, K., & Kumbhar, S. (2023). Encrypted traffic classification using hybrid CNN in SDN. *Future Generation Computer Systems*, 141, 351–362. https://doi.org/10.1016/j.future.2023.03.004

[28] Anwar, F., Hassan, M., & Lee, K. (2024). Edge-intelligent SDN architecture for adaptive traffic flow control. *Journal of Systems Architecture*, 141, 102730. https://doi.org/10.1016/j.sysarc.2024.102730

[29] Gupta, N., Yadav, A., & Banerjee, T. (2025). Cross-layer traffic analysis in SDN using policy-aware flow aggregation. *IEEE Transactions on Network and Service Management*, 22(1), 45–58. https://doi.org/10.1109/TNSM.2025.3145789

[30] Sharma, S., & Kumar, P. (2020). Topology design and traffic analysis in SDN-based networks. *Journal of Network and Computer Applications*, 165, 102720. https://doi.org/10.1016/j.jnca.2020.102720

[31] Al-Fares, M., & Rehman, A. (2020). Impact of network topology on traffic flow in SDN-based architectures. Proceedings of the *IEEE International Conference on Communications* (ICC), 1-6. https://doi.org/10.1109/ICC40277.2020.9149007

[32] Zhang, T., & Li, Y. (2020). Performance evaluation of SDN topology designs for efficient traffic analysis. *IEEE Access,* 8, 50209-50219. https://doi.org/10.1109/ACCESS.2020.2973521

[33] Kaur, M., & Singh, G. (2021). Designing SDN topologies for traffic optimization and scalability in large networks. *Journal of Computing and Communications*, 9(4), 29-39. https://doi.org/10.18576/jcc/090402

[34] Kumar, N., & Pandey, P. (2021). Impact of topology on traffic load distribution in SDN networks. *International Journal of Network Management*, 31(6), e2222. https://doi.org/10.1002/nem.2222

[35] Xiao, J., & Liu, F. (2022). Topology-aware traffic analysis for SDN: A case study in data centers. *IEEE Transactions on Network and Service Management*, 19(1), 42-55. https://doi.org/10.1109/TNSM.2022.3172845

[36] Ahmed, F., & Hussain, F. (2022). Traffic analysis and topology design for resource-efficient SDN-based networking. *Computers*, 11(8), 96. https://doi.org/10.3390/computers11080096

[37] Wang, Y., & Li, X. (2022). Optimizing traffic patterns using topology control in SDN for cloud environments. *Journal of Cloud Computing*: *Advances, Systems and Applications*, 11(1), 22-34. https://doi.org/10.1186/s13677-022-00272-7

[38] Patel, D., & Desai, H. (2023). Topology and traffic load balancing in SDN: A performance comparison study. *Computer Networks*, 214, 108872. https://doi.org/10.1016/j.comnet.2023.108872

[39] Huang, L., & Zhang, Q. (2023). Traffic analysis and optimization techniques for SDN topology in 5G networks. *IEEE Access*, 11, 29345-29358. https://doi.org/10.1109/ACCESS.2023.3262221

[40] Singh, M., & Agarwal, A. (2024). Dynamic traffic analysis in SDN-based hierarchical topologies. *IEEE Transactions on Network and Service Management,* 21(3), 2423-2435. https://doi.org/10.1109/TNSM.2024.3275473

[41] Khan, Z., & Ahmed, M. (2025). Enhancing SDN topology design for optimized traffic routing in smart cities. *Journal of Network and Computer Applications*, 183, 103306. https://doi.org/10.1016/j.jnca.2025.103306

[42] Chatterjee, R., & Das, A. (2021). Multi-threaded controller design for improving flow setup latency in SDN. *Journal of Network Engineering*, 18(2), 101–112.

[43] Lee, J., Kim, S., & Park, H. (2021). Inter-controller communication for distributed SDN environments. *IEEE Transactions on Network and Service Management*, 18(4), 498–510.

[44] Wang, X., & Huang, Y. (2022). Traffic-aware controller scheduling for dynamic load balancing in SDN. *Computer Networks*, 207, 108834.

[45] Sahu, A., Roy, D., & Mukherjee, B. (2022). AI-enabled controller for real-time QoS in SDN. *IEEE Access*, 10, 44756–44770.

[46] Kumar, N., & Singh, R. (2023). Hierarchical SDN control plane for improved fault recovery. *International Journal of Communication Networks and Distributed Systems*, 31(3), 214–229.

[47] Mehmood, A., Asghar, M., & Khan, A. (2023). Deep reinforcement learning-based adaptive routing in SDN. *Future Generation Computer Systems*, 135, 211–223.

[48] Zhou, T., Liu, F., & Chen, Z. (2024). Latency-sensitive scheduling for SDN controllers in 5G. *IEEE Systems Journal*, 18(1), 76–85.

[49] Ali, M., & Rahman, A. (2024). Enhanced load balancing in clustered SDN controllers. *Journal of Network and Computer Applications*, 215, 103542.

[50] Nguyen, L. H., & Patel, S. (2025). Blockchain-based secure control mechanism for multi-domain SDNs. *Computer Communications*, 200, 88–98.

[51] Rana, M., & Iqbal, T. (2025). Predictive fault-tolerant SDN controller using AI. *ACM Transactions on Internet Technology*, 25(2), 1–20.

[52] Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2018). Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 103(1), 14–76. https://doi.org/10.1109/JPROC.2014.2371999

[53] Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2018). A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3), 1617–1634. https://doi.org/10.1109/SURV.2014.012214.00180

10 [54] Kassler, A., Souza, A. M., & Lopes, C. A. (2019). ONOS security extensions: A framework for anomaly detection. *Journal of Network and Systems Management*, 27(2), 312–328. https://doi.org/10.1007/s10922-018-9465-6

10 24 [55] Kim, H., & Feamster, N. (2019). Improving network management with software defined networking. ACM SIGCOMM *Computer Communication Review*, 43(4), 87–92. https://doi.org/10.1145/2534169.2486011

[56] Tootoonchian, A., & Ganjali, Y. (2020). HyperFlow: A distributed control plane for OpenFlow. *IEEE Network*, 34(1), 40–46. https://doi.org/10.1109/MNET.2020.1700174

[57] Arslan, M., Ahmad, I., Hussain, M., & Baig, I. (2020). DynaSDN: A dynamic SDN control framework for improved scalability. *Computer Networks*, 178, 107323. https://doi.org/10.1016/j.comnet.2020.107323

[58] Zhang, C., Wang, Y., & Liu, Z. (2020). Evaluating slicing support in SDN controllers using FlowVisor. *Future Generation Computer Systems*, 112, 755–764. https://doi.org/10.1016/j.future.2020.06.029

[59] Raza, S., & Khokhar, R. H. (2021). FlexiSDN: A control-plane elasticity framework for software-defined WANs. *Journal of Communication Networks*, 23(1), 21–30. https://doi.org/10.1109/JCN.2021.000004

[60] Iqbal, M., Javed, M. Y., & Khan, A. (2021). Real-time performance analysis of software-defined networking controllers. *IEEE Access*, 9, 13456–13466. https://doi.org/10.1109/ACCESS.2021.3052233

[61] Siddiqui, S., Raza, S., & Qamar, F. (2022). SmartSDN: Integrating deep packet inspection into ONOS for enhanced traffic control. *Sensors,* 22(3), 1155. https://doi.org/10.3390/s22031155

[62] Mahmood, A., & Hassan, M. (2022). AIFlow: An intelligent SDN framework for congestion prediction and avoidance. *International Journal of Network Management*, 32(2), e2153. https://doi.org/10.1002/nem.2153

[63] Nguyen, H., Tran, D., & Pham, C. (2022). A comparative performance analysis of software-defined networking controllers. *Computer Networks*, 203, 108620. https://doi.org/10.1016/j.comnet.2021.108620

[64] Thapa, R., & Lee, Y. (2022). QoS-SDN: A real-time SDN framework for bandwidth-aware flow control. *IEEE Transactions on Services Computing*, 15(6), 1103–1112. https://doi.org/10.1109/TSC.2022.3156801

[65] Rahman, A., Alam, M., & Kabir, M. A. (2023). EdgeFlow: An SDN framework for edge computing. *IEEE Internet of Things Journal,* 10(3), 1960–1970. https://doi.org/10.1109/JIOT.2023.3245612

[66] Zhao, F., & Wang, X. (2023). MobSDN: Adaptive SDN for mobile edge networks. *IEEE Vehicular Technology Magazine*, 18(2), 66–73. https://doi.org/10.1109/MVT.2023.3247608

[67] Alzahrani, N., Khan, M. A., & Alotaibi, F. (2023). SecuSDN: A blockchain-based policy enforcement framework for SDN. *IEEE Access*, 11, 44350–44360. https://doi.org/10.1109/ACCESS.2023.3267875

[68] Qureshi, H., & Tariq, M. (2023). GreenSDN: An energy-efficient controller design for sustainable SDN networks. Sustainable Computing*: Informatics and Systems*, 37, 100781. https://doi.org/10.1016/j.suscom.2023.100781

[69] Tan, C., Wong, E., & Lin, J. (2024). AICtrl: A federated learning-enabled SDN controller for smart cloud networks. *Future Internet*, 16(1), 7. https://doi.org/10.3390/fi16010007

[70] Bhardwaj, A., & Kapoor, R. (2024). HybridQoS-SDN: AI-enhanced quality of service optimization for IoT-SDN. *IEEE Sensors Journal*, 24(2), 1347–1356. https://doi.org/10.1109/JSEN.2024.3321674

[71] Chen, J., Liu, H., & Zhou, M. (2024). Benchmarking performance of open-source SDN frameworks under mixed traffic loads. *Computer Standards & Interfaces*, 89, 103773. https://doi.org/10.1016/j.csi.2024.103773

[72] Gupta, R., Singh, P., & Rao, M. (2025). CrossSense: A cross-layer optimization framework for SDN traffic analysis. *IEEE Transactions on Network and Service Management*, 18(1), 112–124. https://doi.org/10.1109/TNSM.2025.3145001

[73] Ahmed, Z., & Sinha, P. (2025). AutoSDN: An autonomous learning-based traffic control framework in SDN. *ACM Transactions on Autonomous Networks*, 11(2), 1–22. https://doi.org/10.1145/3602098

[74] Iqra, F., Rauf, B., & Rehman, U. (2025). FailSafe-SDN: Predictive failure handling in SDN using proactive rerouting. *IEEE Transactions on Reliability*, 74(1), 95–104. https://doi.org/10.1109/TR.2025.3214987

[75] Bai, Y., & Yu, T. (2025). QuantumSDN: Integrating quantum encryption into SDN controller architectures. *Computer Networks*, 238, 110292. https://doi.org/10.1016/j.comnet.2025.110292

[76] Liu, Y., & Zhou, W. (2025). A meta-analysis of SDN controller frameworks: Trends, gaps, and future research. *IEEE Communications Surveys & Tutorials*, 27(2), 85–115. https://doi.org/10.1109/COMST.2025.3285198

[77] Wang, Z., Ding, H., Li, B., Bao, L., Yang, Z., & Liu, Q. (2022). Energy efficient cluster based routing protocol for WSN using firefly algorithm and ant colony optimization. *Wireless Personal Communications*, 125(3), 2167–2200. 23.

[78] Wang, Z., Ding, H., Li, B., Bao, L., & Yang, Z. (2020). An energy efficient routing protocol based on improved artificial bee colony algorithm for wireless sensor networks. *IEEE Access*, 8, 133577–133596. 24.

1 [79] Chica, J. C. C., Imbachi, J. C., & Vega, J. F. B. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, 159(4), 102–118.

1 [80] Bhatia, J., Dave, R., Bhayani, H., Tanwar, S., & Nayyar, A. (2020). Sdn-based real-time urban traffic analysis in vanet environment. *Computer Communications*, 149, 162–175. 35.

31 [81] Priya, A. V., & Radhika, N. (2019). Performance comparison of SDN OpenFlow controllers. *International Journal of Computer Aided Engineering and Technology*, 11(4–5), 467–479.

39 [82] Leon, J., Aydeger, A., Mercan, S., & Akkaya, K. (2023). SDN-enabled vehicular networks: Theory and practice within platooning applications. *Vehicular Communications*, 39, 100545. 48.

25 [83] Singh, P. K., Sharma, S., Nandi, S. K., & Nandi, S. (2019). Multipath TCP for V2I communication in SDN controlled small cell deployment of smart city. *Vehicular communications,* 15, 1–15.

21 [84] Rezaee, M. R., Hamid, N. A. W. A., Hussin, M., & Zukarnain, Z. A. (2024). Fog Offloading and Task Management in IoT-Fog-Cloud Environment: Review of Algorithms, Networks and SDN Application. *IEEE Access*.

36 [85] Ayodele, B., & Buttigieg, V. (2024). SDN as a defense mechanism: a comprehensive survey. *International Journal of Information Security*, 23(1), 141-185.

8 [86] Qaffas, A. A., Kamal, S., Sayeed, F., Dutta, P., Joshi, S., & Alhassan, I. (2023). Adaptive population-based multi-objective optimization in SDN controllers for cost optimization. *Physical Communication*, 58, 102006.

35 [87] Wang, K., Fu, Y., Duan, X., Liu, T., & Xu, J. (2024). An abnormal traffic detection system in SDN is based on deep learning hybrid models. *Computer Communications,* 216, 183-194.

4 [88] Li, J., Qi, X., Li, J., Su, Z., Su, Y., & Liu, L. (2024). Fault Diagnosis in the Network Function Virtualization: A Survey, Taxonomy and Future Directions. *IEEE Internet of Things Journal.*

37 [89] Ramya, G., & Manoharan, R. (2023). Traffic-aware dynamic controller placement in SDN using NFV. *The Journal of Supercomputing*, 79(2), 2082-2107.

26 [90] Núñez-Gómez, C., Carrión, C., Caminero, B., & Delicado, F. M. (2023). S HIDRA: A blockchain and SDN domain-based architecture to orchestrate fog computing environments. *Computer Networks*, 221, 109512.

16 [91] Song, S., Park, H., Choi, B. Y., Choi, T., & Zhu, H. (2017). Control path management framework for enhancing software-defined network (SDN) reliability. *IEEE Transactions on Network and Service Management*, 14(2), 302-316.

23 [92] Tang, D., Zheng, Z., Yin, C., Xiong, B., Qin, Z., & Yang, Q. (2024). FTODefender: An efficient flow table overflow attacks defending system in SDN. *Expert Systems with Applications*, 237, 121460.

# 7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

## Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 8 words)

## Match Groups

- 228 Not Cited or Quoted 7%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

## Top Sources

| | |
|---|---|
| 4% | Internet sources |
| 4% | Publications |
| 5% | Submitted works (Student Papers) |

## Integrity Flags

**0 Integrity Flags for Review**

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.