AI-Based Automatic Network Traffic Routing To Avoid Congestion

Someshwaran S[1], Ilayalatha S[2]
[1]Assist prof, Dept of Computer Science And Engineering,
[2]Dept Of Computer Science And Engineering,
[1, 2]AVS ENGINEERING COLLEGE, SALEM, TAMIL NADU

Abstract- *The exponential increase in network traffic has made congestion a major challenge, leading to increased latency, packet loss, and reduced Quality of Service (QoS). This paper presents an intelligent and adaptive traffic routing system that integrates Software-Defined Networking (SDN), Python-based automation, and Artificial Intelligence/Machine Learning (AI/ML) to proactively detect and mitigate network congestion. The proposed system utilizes the centralized control capability of SDN to continuously monitor network conditions and dynamically manage traffic flows. Machine Learning algorithms are trained using both historical and real-time network data to accurately predict congestion based on key performance metrics such as bandwidth utilization, delay, and packet loss. Upon identifying potential congestion, the system automatically selects optimal alternative paths and reroutes traffic in real time, ensuring efficient bandwidth utilization and reduced network delays. Experimental evaluation shows that the integration of AI/ML with SDN significantly improves throughput, reduces congestion levels, and achieves effective load balancing. This research demonstrates a smart and scalable approach for next-generation networks, suitable for data centers, cloud computing, and IoT infrastructures.*

Keywords: Software-Defined Networking (SDN), Network Congestion, Machine Learning, Traffic Routing, QoS, Python Automation, AI/ML, Load Balancing

I. INTRODUCTION

In the modern digital era, the exponential growth of internet usage, cloud-based services, video streaming platforms, and Internet of Things (IoT) devices has resulted in a massive surge in network traffic. As organizations and individuals increasingly rely on real-time applications such as video conferencing, online gaming, and critical data services, maintaining efficient and reliable network performance has become essential.

Network congestion significantly degrades system performance, resulting in increased latency, higher packet loss, reduced throughput, and overall deterioration of Quality of Service (QoS). Traditional network architectures primarily depend on static routing protocols such as OSPF and RIP, which are designed based on predefined rules and fixed configurations. While these protocols perform adequately under stable conditions, they lack the intelligence and adaptability required to handle dynamic and unpredictable traffic patterns.

To overcome these challenges, Software-Defined Networking (SDN) has emerged as a revolutionary networking paradigm. SDN introduces a clear separation between the control plane and the data plane, enabling centralized control and programmability of the network. Building upon SDN capabilities, this paper proposes an advanced traffic routing system that integrates Python-based automation and Artificial Intelligence/Machine Learning (AI/ML) techniques.

Machine Learning models are trained using both historical and real-time network data to identify patterns and trends in traffic behavior, enabling the system to predict potential congestion scenarios before they occur. Once a potential congestion is identified, the system dynamically computes alternative optimal paths and reroutes traffic accordingly, providing greater scalability, adaptability, and resilience.

II. ANALYSIS OF EXISTING SYSTEMS

In conventional networking environments, traffic routing is managed using standard routing protocols such as OSPF, RIP, and BGP. These protocols select optimal paths for data transmission based on predefined parameters like hop count, link cost, or shortest path. However, once configured, these routing decisions tend to remain static and are not capable of adapting dynamically to fluctuating network conditions.

Traditional network management follows a distributed architecture, where each router independently determines routing decisions based on its limited, local view of the network. This lack of centralized control restricts the

ability to analyze the overall network state, thereby reducing the efficiency of congestion detection and management.

2.1 Disadvantages of Existing Systems

- **Limited Dynamic Adaptability:** Traditional routing protocols are not designed to handle rapidly changing traffic conditions, making them inefficient in dynamic network environments.
- **Absence of Centralized Control:** The distributed nature of conventional networks prevents a unified, global view, limiting effective traffic optimization and management.
- **Reactive Approach to Congestion:** Congestion is addressed only after it occurs, leading to performance degradation and poor Quality of Service (QoS).
- **Dependence on Manual Intervention:** Network configuration and maintenance require continuous human involvement, increasing operational complexity and the likelihood of errors.
- **Poor Resource Utilization:** Fixed routing paths often lead to uneven traffic distribution, causing some links to become congested while others remain underutilized.
- **Scalability Issues:** As network size and traffic demands grow, traditional systems struggle to maintain efficiency and performance.
- **Lack of Predictive Capability:** The absence of data-driven analysis and machine learning techniques prevents early detection and prevention of congestion.

III. PROPOSED SYSTEM

To address the limitations of conventional networking approaches, this paper introduces an intelligent and automated traffic routing framework that integrates Software-Defined Networking (SDN), Python-based automation, and Artificial Intelligence/Machine Learning (AI/ML) techniques. The proposed system is built upon the SDN architecture, which separates the control plane from the data plane to enable centralized network management.

The SDN controller maintains a comprehensive, real-time view of the entire network, continuously monitoring critical parameters such as bandwidth utilization, latency, packet loss, and traffic flow patterns. Python is employed to develop automation modules that interact seamlessly with the SDN controller via APIs. A key component of the proposed system is the AI/ML-based prediction model, trained using both historical and real-time network data to identify traffic patterns and forecast potential congestion scenarios.

Based on these predictions, the system dynamically determines optimal routing paths and reroutes traffic away from congested links. This proactive routing strategy ensures efficient utilization of available network resources, reduces transmission delays, and enhances overall network performance. The system also incorporates intelligent load balancing by distributing traffic evenly across multiple paths.

3.1 Advantages of the Proposed System

- **Real-Time Dynamic Traffic Management:** The system continuously adapts to changing network conditions, enabling efficient and flexible routing decisions.
- **Centralized Control and Global Visibility:** SDN provides a unified view of the network, allowing better monitoring, control, and optimization of traffic flows.
- **Proactive Congestion Prevention:** AI/ML models predict congestion in advance, reducing packet loss, delay, and improving QoS.
- **Automation and Reduced Human Intervention:** Python-based automation minimizes manual configuration, reducing operational effort and human errors.
- **Optimized Resource Utilization:** Traffic is intelligently distributed across multiple paths, ensuring balanced usage of network resources.
- **High Scalability and Flexibility:** The system can easily adapt to growing network sizes and increasing traffic demands.

IV. SYSTEM ARCHITECTURE AND MODULE DESIGN

The proposed AI-based network traffic routing system is designed using a modular architecture, where each module performs a specific function while working collaboratively with other components. This modular design improves system flexibility, scalability, and ease of implementation.

4.1 Network Monitoring Module

The Network Monitoring Module plays a crucial role in maintaining real-time awareness of network conditions. It continuously observes and collects data from various network devices such as switches and routers within the SDN environment. By interacting with the SDN controller, this module gathers essential performance metrics including bandwidth utilization, latency, packet loss, throughput, and traffic flow statistics.

4.2 Data Collection and Preprocessing Module

This module is responsible for transforming raw network data into a structured and usable format for analysis. It performs preprocessing steps including data cleaning, filtering, normalization, and feature extraction. Data cleaning involves removing errors and handling missing values, while normalization ensures all parameters are scaled uniformly for better model performance.

4.3 Machine Learning Prediction Module

The Machine Learning Prediction Module acts as the intelligence layer of the system. It utilizes advanced AI/ML algorithms — such as Decision Trees, Random Forest, Support Vector Machines, or Neural Networks — to analyze both historical and real-time network data. By learning from past traffic patterns and trends, the model is capable of identifying potential congestion scenarios before they occur.

4.4 Decision-Making Module

The Decision-Making Module is responsible for selecting the most efficient routing strategy once congestion is predicted. It evaluates multiple alternative network paths by considering key performance factors such as delay, bandwidth availability, link utilization, and congestion levels, then determines the optimal path for smooth and efficient data transmission.

4.5 Traffic Routing and Control Module

This module directly communicates with the SDN controller to modify routing policies and update flow tables in network switches. By dynamically adjusting traffic paths, it ensures that data packets are routed through less congested and more efficient links. It also supports load balancing by distributing traffic across multiple paths.

4.6 Python-Based Automation Module

The Automation Module integrates all system components and ensures seamless coordination between them. Implemented using Python, this module automates key processes such as data collection, preprocessing, communication with the SDN controller, execution of machine learning models, and routing updates, reducing manual intervention and minimizing human errors.

4.7 Visualization and Monitoring Module

This module provides an interactive interface for network administrators to observe and analyze system performance. It displays important metrics such as network traffic, congestion levels, bandwidth usage, and routing paths in the form of graphs, charts, or dashboards, enhancing transparency and enabling real-time performance evaluation.

V. SYSTEM DESIGN

5.1 Software Environment

The system is implemented using Python as the primary programming language due to its simplicity, extensive libraries, and strong support for networking and data analysis. Python provides robust frameworks for machine learning (e.g., scikit-learn, TensorFlow) and SDN integration via REST APIs. The development environment utilizes Anaconda as the GUI tool and runs on Windows 10.

5.2 Hardware Requirements

The system operates on the following minimum hardware configuration: Intel Core i3 processor at 2.3 GHz, 4 GB RAM, 500 GB hard disk, standard keyboard and three-button mouse, and a 17-inch LED monitor.

5.3 Input and Output Design

The input design serves as the link between the information system and the user. It encompasses specifications and procedures for data preparation, focusing on controlling the amount of input required, avoiding errors, maintaining simplicity, and ensuring security and privacy. Input parameters include real-time network metrics such as bandwidth utilization, delay, jitter, and packet loss collected from SDN-connected devices.

Output design communicates processed information to end users, conveying past activities, current network status, and future projections. It signals important events such as congestion warnings, triggers automated rerouting actions, and confirms routing decisions through dashboards and visualization components.

VI. TESTING AND IMPLEMENTATION

Testing was conducted to verify that all system components meet their functional requirements and operate correctly. Multiple testing strategies were employed to ensure system reliability and correctness.

- **Unit Testing:** Individual modules were tested independently to validate internal logic, ensuring that program inputs produce valid outputs and all decision branches execute correctly.
- **Integration Testing:** Integrated software components were tested to confirm that individual modules work correctly as a combined system, exposing any interface or communication problems.
- **Functional Testing:** Systematic demonstrations confirmed that valid inputs are accepted, invalid inputs are rejected, and all system functions operate as specified by technical requirements.
- **White Box Testing:** Internal logic paths, code structure, and program flow were examined to validate areas unreachable through black-box methods.
- **Black Box Testing:** The system was tested as a whole without knowledge of internal workings, verifying that correct outputs are produced for given inputs.
- **Acceptance Testing:** End-user acceptance testing confirmed that the system meets all functional requirements and satisfies user expectations.

All test cases passed successfully. No critical defects were encountered during the testing phase.

VII. FEASIBILITY STUDY

7.1 Economic Feasibility

The economic feasibility study confirms that the project can be developed within budget constraints. Most technologies used — including Python, SDN controllers such as Ryu, and the Mininet emulator — are freely available as open-source software. Only minimal hardware investment is required.

7.2 Technical Feasibility

The technical requirements of the system are modest, requiring only standard computing hardware and readily available open-source software tools. The system integrates well-established networking frameworks and machine learning libraries, making implementation straightforward with minimal technical risk.

7.3 Social Feasibility

The system is designed to be user-friendly and easy to adopt. Network administrators interact primarily through a visual dashboard, reducing the learning curve. The automation of routine tasks reduces operational complexity, and the system's ability to self-optimize increases administrator confidence and acceptance.

VIII. CONCLUSION

This paper has presented an intelligent and automated approach for network traffic routing to address the critical issue of network congestion. By integrating Software-Defined Networking (SDN) with Python-based automation and AI/ML techniques, the proposed system provides a modern and efficient solution for dynamic traffic management.

Unlike traditional networking methods that rely on static routing and reactive congestion handling, the proposed system adopts a proactive approach by continuously monitoring network conditions and predicting congestion in advance. The centralized control offered by SDN enhances network visibility and simplifies management, while Python-based automation reduces manual intervention and improves system responsiveness.

The modular design ensures scalability and flexibility, making it suitable for deployment in data centers, enterprise networks, cloud infrastructures, and IoT ecosystems. Results demonstrate that the integration of AI/ML with SDN significantly enhances network efficiency and reliability, presenting a smart, adaptive, and future-ready solution for next-generation network traffic management.

8.1 Future Scope

- **Advanced AI/ML Integration:** Future work can incorporate deep learning techniques such as DNN, RNN, and Reinforcement Learning to enhance congestion prediction accuracy and enable more intelligent, adaptive routing decisions.
- **Real-Time Large-Scale Deployment:** The system can be extended for real-time implementation in large-scale environments like cloud data centers, 5G networks, and smart cities.
- **Enhanced Security and Automation:** Integration of AI-based threat detection and self-healing mechanisms can improve network security and enable autonomous network management with minimal human intervention.

REFERENCES

[1] T. Nadeau and K. Gray, SDN: Software Defined Networks, O'Reilly Media, 2013.

[2] A. Lara, A. Kolasani, and B. Ramamurthy, "Network Innovation using OpenFlow: A Survey," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 493–512, 2014.
[3] N. McKeown et al., "OpenFlow: Enabling Innovation in Campus Networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.
[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.
[5] S. Haykin, Neural Networks and Learning Machines, 3rd ed., Pearson, 2009.
[6] Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," ONF White Paper, 2012.
[7] D. Kreutz et al., "Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, 2015.
[8] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," IEEE Communications Magazine, vol. 51, no. 2, pp. 114–119, 2013.
[9] T. Chen, S. Zhang, and Y. Li, "Machine Learning in Network Traffic Prediction: A Survey," IEEE Access, vol. 6, pp. 22320–22335, 2018.
[10] A. Mestres et al., "Knowledge-Defined Networking," ACM SIGCOMM Computer Communication Review, vol. 47, no. 3, pp. 2–10, 2017.
[11] W. Xia et al., "A Survey on Software-Defined Networking," IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27–51, 2015.
[12] Y. Li et al., "Deep Learning for Network Traffic Prediction: A Review," IEEE Communications Surveys & Tutorials, 2020.
[13] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2016.
[14] J. Moy, "OSPF Version 2," IETF RFC 2328, 1998.
[15] Mininet Project, "An Instant Virtual Network on your Laptop," Available: http://mininet.org/
[16] Ryu SDN Framework, "Ryu Network Operating System," Available: https://osrg.github.io/ryu/