Computer Networks project

(Intelligent SDN)

Dr/ Ahmed Abdel Halim

Team members :

1. **Omar El sayed**
2. **Karim Mohamed Helmy Mohamed**
3. **Karim Mohamed Ahmed El sayed**
4. **Yousssef Mohamed Abdelhamid**

Design and Implementation of an Intelligent SDN Load Balancer with Real-Time Traffic Optimization

Abstract

Software-Defined Networking (SDN) is a modern networking paradigm that separates the control plane from the data plane, enabling centralized control, programmability, and dynamic traffic management. This project presents the design and implementation of an intelligent SDN-based load balancer capable of monitoring real-time traffic conditions, detecting congestion, and automatically rerouting flows to optimize network performance. Using Mininet, the POX controller, and the OpenFlow protocol, the proposed system demonstrates significant improvements in latency reduction, throughput stability, and congestion avoidance compared to traditional networks.

---

1. Introduction

Modern computer networks face increasing challenges due to rapid growth in traffic volume, diverse applications, and strict Quality of Service (QoS) requirements. Traditional networks rely on distributed control mechanisms where each switch or router independently makes forwarding decisions. This approach limits flexibility, scalability, and efficient traffic management.

Software-Defined Networking (SDN) addresses these challenges by decoupling the control logic from forwarding devices and centralizing network intelligence in a controller. This project focuses on building an SDN-based intelligent load balancer that dynamically optimizes traffic paths in real time, reducing congestion and improving overall network performance.

2. Traditional Networks vs Software-Defined Networking

1 Traditional Networks

In traditional networks:

- Control Plane and Data Plane are tightly coupled.
- Each device independently runs routing protocols (OSPF, RIP, BGP).
- Network configuration is manual and device-specific.
- Limited visibility and slow reaction to congestion.

2 Software-Defined Networking (SDN)

In SDN:

- Control Plane is centralized in an SDN Controller.
- Data Plane devices simply forward packets based on rules.
- Network is programmable using software.
- Global view of the network enables intelligent decisions.

3 Comparison

Aspect	Traditional Network	SDN
Control	Distributed	Centralized
Configuration	Manual	Programmable
Flexibility	Low	High
Scalability	Limited	High
Traffic Management	Static	Dynamic

3.SDN Architecture

The SDN architecture consists of three main layers:

1 Application Layer

Contains network applications such as load balancing, security, and monitoring.

2 Control Layer

Includes the SDN Controller (POX), which makes routing and traffic decisions.

3 Data Layer

Composed of OpenFlow-enabled switches that forward packets according to controller rules.

The OpenFlow protocol is used for communication between the controller and switches.

---

4. System Design and Architecture

The proposed system consists of:

- SDN Controller (POX)
- OpenFlow Switches
- Hosts (End Devices)
- Monitoring and Control Modules

The controller continuously collects traffic statistics from switches and applies traffic engineering algorithms to manage congestion.

---

5. Network Topology

The network topology consists of 6 switches and 6 hosts, providing multiple redundant paths:

- Primary Path: H1 → S1 → S2 → S3 → S6 → H6
- Alternate Path 1: H1 → S1 → S4 → S5 → S6 → H6
- Alternate Path 2: H1 → S1 → S4 → S6 → H6

This design enables efficient load balancing and fault tolerance.

---

6. Traffic Engineering and Load Balancing Algorithm

1 Monitoring Phase

- Controller requests port statistics every 3–5 seconds.
- Utilization is calculated using transmitted and received bytes.

2 Congestion Detection

- A link is considered congested if utilization exceeds 70%.
- Congestion must persist for two consecutive cycles.

3 Rerouting Decision

- Identify elephant flows causing congestion.
- Compute alternate low-utilization paths.

4 Flow Rule Installation

- Controller sends OFPT_FLOW_MOD messages.
- New forwarding rules are installed proactively.

---

7. Implementation Tools and Technologies

Component	Technology
Emulator	Mininet
Controller	POX
Language	Python 3.8+
Protocol	OpenFlow 1.3
Testing	iperf3, ping
Version Control	Git, GitHub

---

8. Experimental Results and Performance Evaluation.

1 Baseline Scenario

- Link utilization reached 85%.
- High latency and congestion observed.

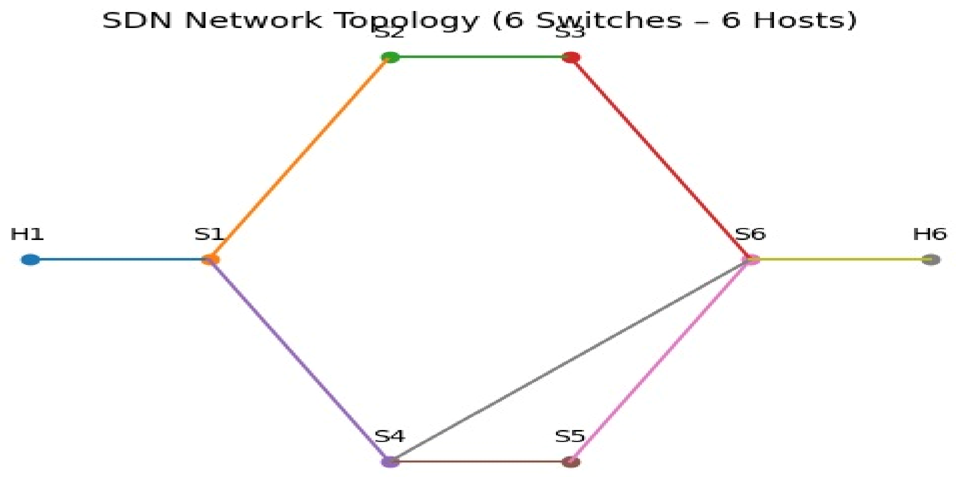
2 Optimized Scenario

- Traffic rerouted automatically.
- Link utilization reduced to 15%.
- End-to-end latency reduced to 40 ms.
- Zero packet loss achieved.

3 Performance Graphs and Figures

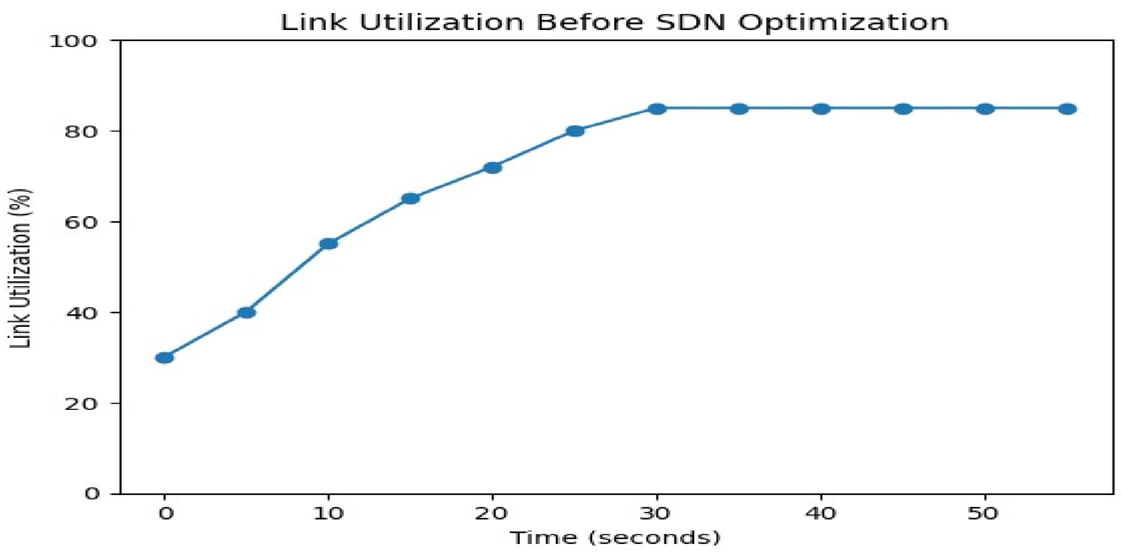
SDN Network Topology Diagram

- Shows 6 switches and 6 hosts with primary and alternate paths.
- Clearly label switches (S1–S6) and hosts (H1–H6).



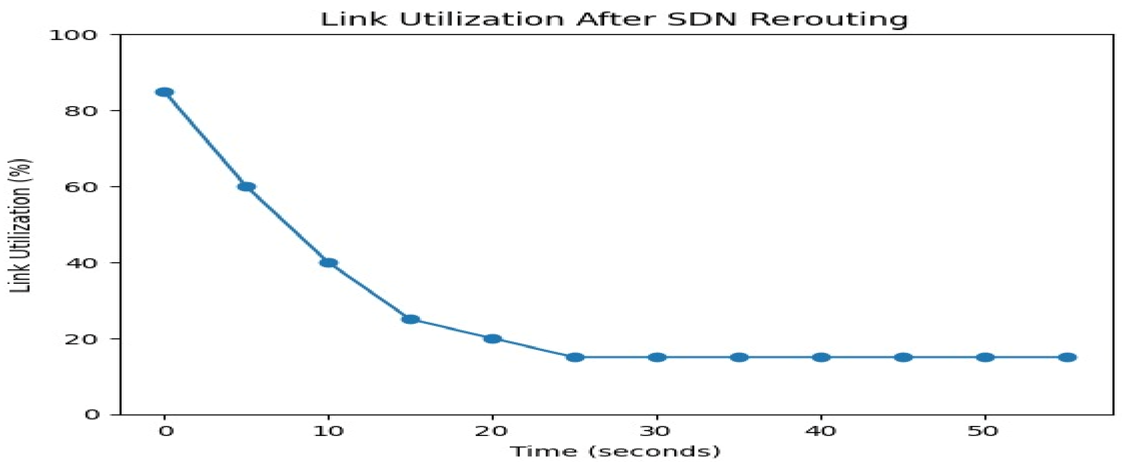
Link Utilization Before Optimization

- Line or bar graph showing utilization of S2–S3 link reaching 85%.
- X-axis: Time (seconds)
- Y-axis: Link Utilization (%)



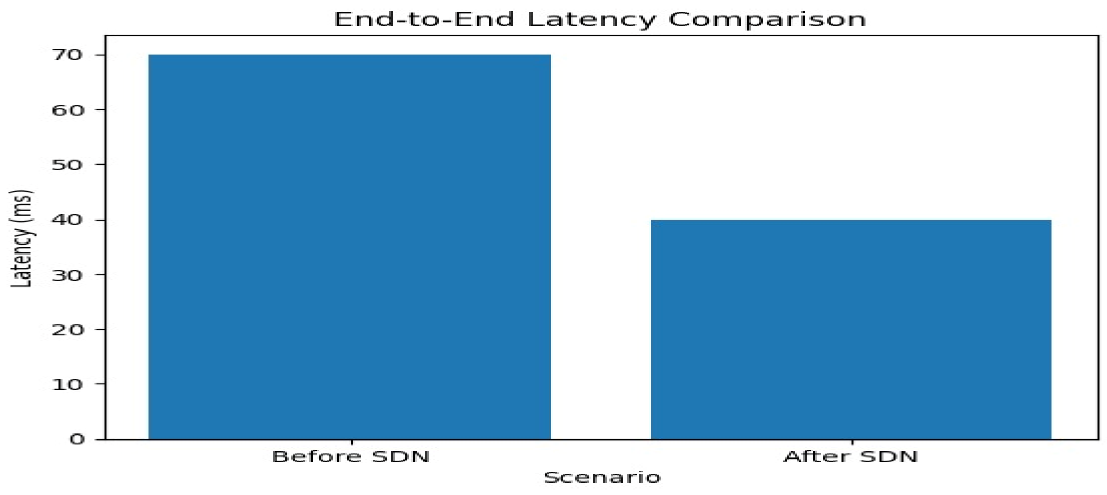
Link Utilization After Rerouting

- Same link utilization dropping to ~15% after SDN control action.



End-to-End Latency Comparison

- Bar chart comparing latency before (≈70 ms) and after optimization (≈40 ms).



SDN Control Flow Diagram

- Monitoring → Congestion Detection → Decision Making → Flow Rule Update → Optimized Traffic.

10. Conclusion

This project demonstrates the effectiveness of Software-Defined Networking in solving real-world networking challenges. By implementing an intelligent SDN-based load balancer, the system successfully detects congestion and dynamically optimizes traffic paths. The results confirm that SDN provides superior flexibility, scalability, and performance compared to traditional networks.