	Helwan National University

Course Title **Computer Networks**	**Project**
Course Coordinator **Prof. Ahmed M. Abdelhaleem**	Semester **one 25-26**

Project Title

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

*Submission Deadline**: [25 December 2025]***

*Group Size**: Up to 5 students***

1. Project Overview

This 8-week project introduces students to Software-Defined Networking (SDN) — a modern approach to managing and controlling computer networks. Students will work in small teams to design and implement a system that can balance network traffic automatically, reducing congestion and improving performance. By completing this project, students will gain hands-on experience with SDN principles, controller operation, and real-time traffic optimization.

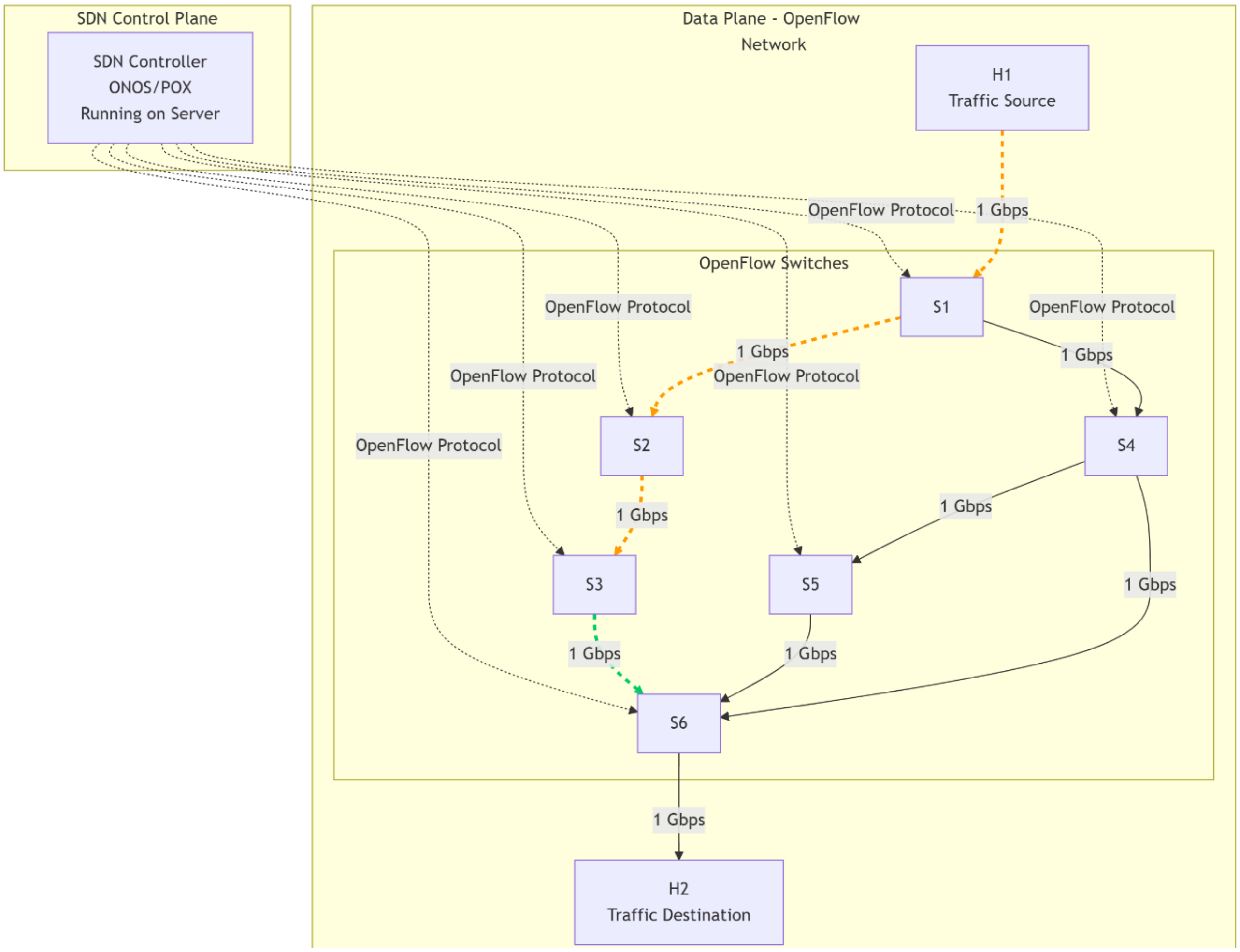
2. Learning Objectives

- Explain the structure and working of an SDN network and the role of the OpenFlow protocol.
- Use an SDN controller (such as POX) to develop and test basic applications.
- Build a system that can monitor real-time traffic and detect congestion.
- Create a traffic management algorithm that reroutes data when needed.
- Measure and analyze how the system improves performance.

3. System Architecture

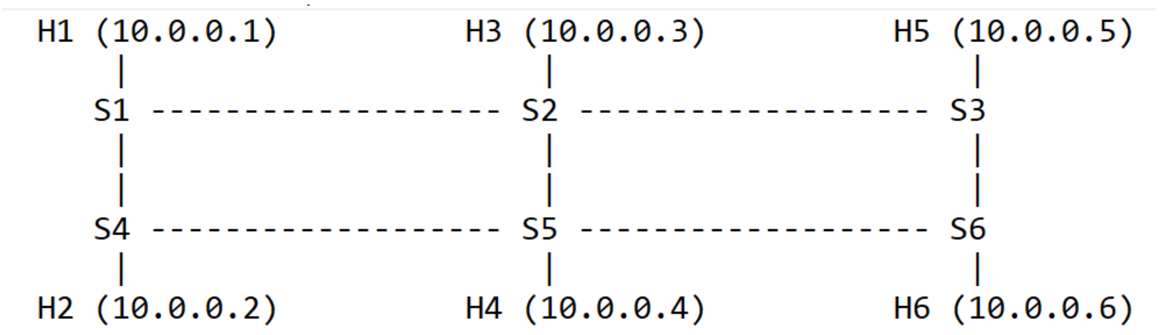
The SDN model separates the control plane (the "brain") from the data plane (the "muscle"). Key components include:

- SDN Controller (Control Plane): Software that decides how data should flow through the network (e.g., POX, ONOS).
- OpenFlow Switches (Data Plane): Devices that follow the controller's instructions to forward packets.
- Hosts (End Devices): Computers or servers that send and receive data.
- OpenFlow Protocol: The communication method between controller and switches.



4. Network Topology Design

The project uses a network of 6 switches and 6 hosts, providing multiple routes between source and destination. This setup allows testing of dynamic routing, load balancing, and fault tolerance.



```
H1 (10.0.0.1)          H3 (10.0.0.3)          H5 (10.0.0.5)
     |                      |                      |
    S1 ------------------- S2 ------------------- S3
     |                      |                      |
     |                      |                      |
    S4 ------------------- S5 ------------------- S6
     |                      |                      |
H2 (10.0.0.2)          H4 (10.0.0.4)          H6 (10.0.0.6)
```

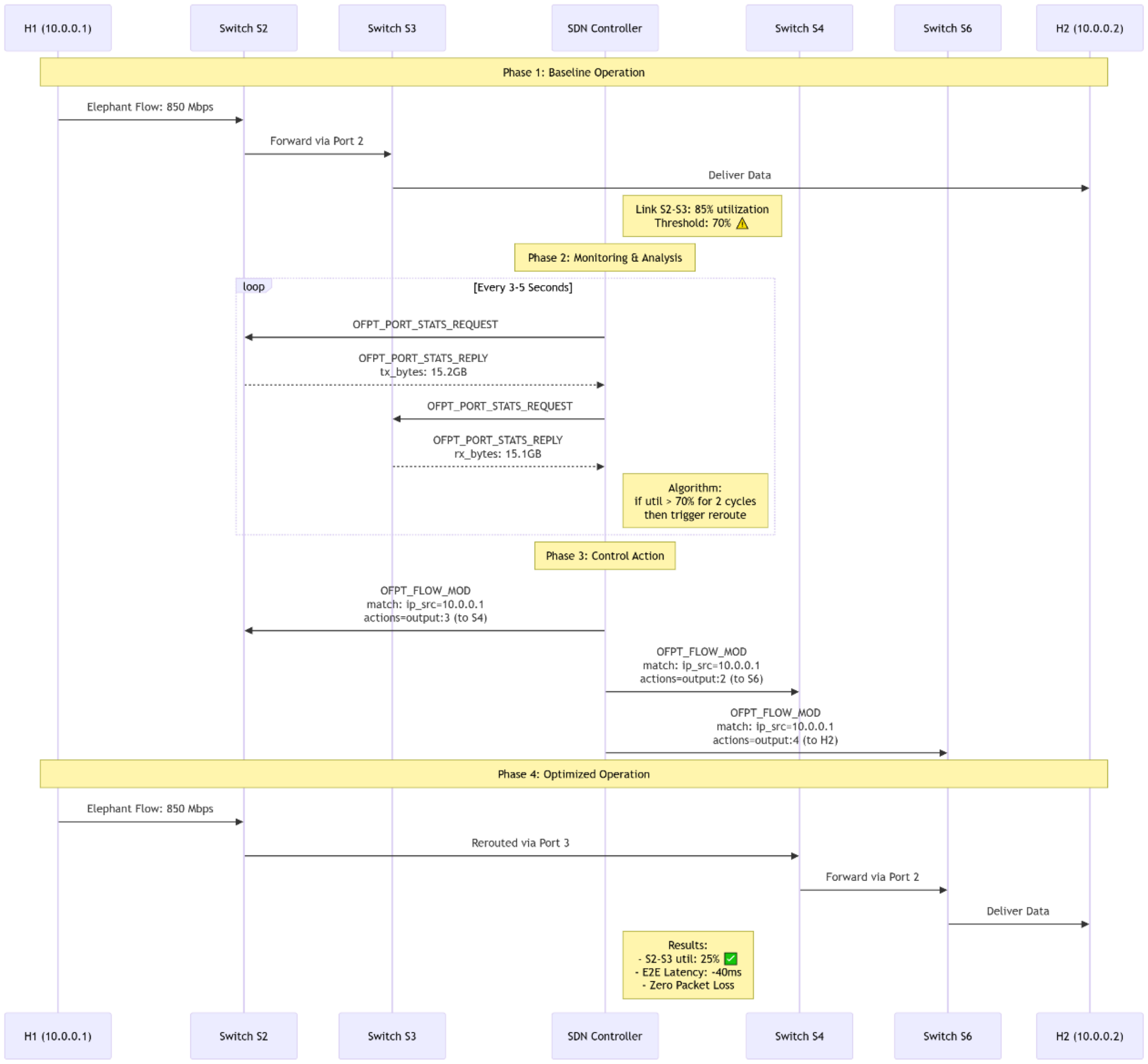
- Primary Path: H1 → S1 → S2 → S3 → S6 → H6
- Alternate Path 1: H1 → S1 → S4 → S5 → S6 → H6
- Alternate Path 2: H1 → S1 → S4 → S6 → H6

5. Traffic Engineering Algorithm

The traffic engineering algorithm monitors link utilization and automatically reroutes large data flows to avoid congestion. It follows these main steps:

1. Monitor network links regularly (every few seconds).
2. Detect congestion if utilization exceeds a threshold (e.g., 70%).
3. Select large data flows causing congestion.
4. Find alternate, less busy paths.
5. Update switch flow rules using the OpenFlow protocol.

Example: If the link between S2 and S3 is overloaded (85%), the controller reroutes traffic via S2 → S4 → S6. This reduces congestion and maintains smooth data flow.



H1 (10.0.0.1)
Switch S2
Switch S3
SDN Controller
Switch S4
Switch S6
H2 (10.0.0.2)
Phase 1: Baseline Operation
Elephant Flow: 850 Mbps
Forward via Port 2
Deliver Data
Link S2-S3: 85% utilization Threshold: 70% ⚠
Phase 2: Monitoring & Analysis
loop
[Every 3-5 Seconds]
OFPT_PORT_STATS_REQUEST
OFPT_PORT_STATS_REPLY tx_bytes: 15.2GB
OFPT_PORT_STATS_REQUEST
OFPT_PORT_STATS_REPLY rx_bytes: 15.1GB
Algorithm: if util > 70% for 2 cycles then trigger reroute
Phase 3: Control Action
OFPT_FLOW_MOD match: ip_src=10.0.0.1 actions=output:3 (to S4)
OFPT_FLOW_MOD match: ip_src=10.0.0.1 actions=output:2 (to S6)
OFPT_FLOW_MOD match: ip_src=10.0.0.1 actions=output:4 (to H2)
Phase 4: Optimized Operation
Elephant Flow: 850 Mbps
Rerouted via Port 3
Forward via Port 2
Deliver Data
Results: - S2-S3 util: 25% ✅ - E2E Latency: -40ms - Zero Packet Loss

<u>Phase 1: Baseline Operation (Congestion Detected)</u>

Initially, a large 850 Mbps "elephant flow" is traveling from host H1 (10.0.0.1) to host H2 (10.0.0.2). The traffic is routed through switches S2 and S3. This high volume of traffic overloads the link between S2 and S3, causing its utilization to spike to 85%, which is well above the pre-defined 70% threshold.

<u>Phase 2: Monitoring & Analysis</u>

The SDN Controller, the network's brain, is constantly monitoring the network's health.

- Data Collection: In a loop (e.g., every 3-5 seconds), it sends OFPT_PORT_STATS_REQUEST messages to the switches using the OpenFlow protocol.
- Data Analysis: The switches (like S3) respond with their current traffic statistics (tx_bytes, rx_bytes).
- Decision Making: The controller's algorithm detects that the link utilization has remained above the 70% threshold for two consecutive monitoring cycles. This triggers a decision to reroute the flow to alleviate the congestion.

<u>Phase 3: Control Action (Rerouting)</u>

Having decided on a new, less congested path, the controller takes action. It sends OFPT_FLOW_MOD (Flow Modification) messages to proactively install new forwarding rules on the relevant switches:

- It tells Switch S2 to send traffic from H1 to Switch S4.
- It tells Switch S4 to forward that traffic to Switch S6.
- It tells Switch S6 to forward the traffic to its final destination, H2.

This effectively creates a new path for the elephant flow: S2 → S4 → S6 → H2.

<u>Phase 4: Optimized Operation (Problem Solved)</u>

The elephant flow is now seamlessly rerouted along the new path. The results demonstrate the success of the operation:

- Congestion Resolved: The utilization on the original S2-S3 link drops to a healthy 15%.
- Good Performance: The flow maintains a low end-to-end latency of 40ms with zero packet loss.

6. Project Plan

Week	Focus Area	Tasks	Deliverables
1–2	Setup & Basics	Install Mininet and POX, create simple network, test connectivity, set up GitHub.	Working basic network.
3–4	Monitoring Module	Collect switch statistics, calculate link utilization, detect congestion.	Live monitoring system.
5–6	Control Logic	Implement rerouting algorithms, manage flow rules and priorities.	Automatic rerouting system.
7–8	Testing & Optimization	Tune performance, measure latency, demonstrate results.	Final demo and report.

Project Roadmap: Here's how you'll build this project, week by week.

Weeks 1-2: Building the Foundation. You'll set up your tools (Mininet, POX) and create the virtual network world where your application will live.

Weeks 3-4: Creating the "Eyes". You'll write the monitoring part of your code. This module will watch the network and detect when a traffic jam occurs.

Weeks 5-6: Programming the "Brain". This is where you'll implement the control logic. Your code will learn how to automatically calculate a new path and send the commands to reroute traffic.

Weeks 7-8: Optimizing and Perfecting. You'll fine-tune your algorithm, run final tests, and prepare your awesome demo to show off how well your system works.

7. Tools & Technologies

Component	Technology	Purpose
Network Emulator	Mininet	Build virtual network topology
SDN Controller	POX	Manage and control traffic
Programming Language	Python 3.8+	Develop controller applications
Protocols & APIs	OpenFlow 1.3, REST API	Controller-switch communication
Testing Tools	iperf3, ping	Performance measurement
Version Control	Git, GitHub	Manage files and teamwork

8. Deliverables

Weekly Checkpoints:

- Week 2 – Network setup and topology diagram
- Week 4 – Monitoring module with live data
- Week 6 – Working automatic rerouting
- Week 8 – Final system and performance analysis

Final Submission: Source code, report (8–10 pages), demo video, and presentation.

9. Evaluation Criteria

Category	Weight	Description
Functionality	40%	System performs all required tasks successfully
Code Quality	20%	Clean, readable, and modular code
Performance	15%	Improved latency and throughput
Documentation	15%	Clear and complete report
Presentation	10%	Professional and understandable delivery

10. Success Indicators

- Detect congested links within 10 seconds.
- Reroute traffic automatically with zero packet loss.
- Reduce latency by at least 40%.
- Balance load across available links.

11. References

- Open Networking Foundation. 'OpenFlow Switch Specification'
- POX Controller Documentation
- Mininet Official Documentation
- Kreutz et al., 'Software-Defined Networking: A Comprehensive Survey'