MICRO PROJECT

ADVANCED COMPUTER NETWORKS (M24CS1E105A)

Simulating SRv6-Based SD-WAN QoS Management

MICRO PROJECT REPORT

Submitted by

PAUL JOSE

MAC24CSCE07

To

The APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY in partial fulfillment for the award of the degree

of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
(GOVT. AIDED & AUTONOMOUS)
KOTHAMANGALAM, KERALA-686 666

DECEMBER 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING (GOVT.AIDED AUTONOMOUS)
KOTHAMANGALAM, KERALA-686 666

CERTIFICATE

This is to certify that the report entitled **“Simulating SRv6-Based SD-WAN QoS Management”** submitted by **Mr. Paul Jose , Reg No: MAC24CSCE07** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Computer Science & Engineering for the academic year 2024-2026 is a bonafied record of the micro project presented by them under our supervision and guidance. This report in any form has not been submitted to any other university or Institute for any purpose.

....

Prof. Shabiya M I
Staff in Charge

....

Prof. Joby George
Head of the Department

ACKNOWLEDGEMENT

First and foremost, I sincerely thank **God Almighty** for his grace for the successful and timely completion of the micro project. I express my sincere gratitude and thanks to the Principal **Dr. Bos Mathew Jos** and Head of the Department **Prof. Joby George** for providing the necessary facilities and their encouragement and support. I owe special thanks to the faculty in charge **Prof. Shabiya M I** for their corrections, suggestions and efforts to coordinate the micro project under a tight schedule. I also express my gratitude to the staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to completing this microproject. Finally, I would like to acknowledge the tremendous support given to me by our dear friends without whose support this work would have been all the more difficult to accomplish.

ABSTRACT

The shift toward Software-Defined Wide Area Networks (SD-WAN) utilizing Segment Routing over IPv6 (SRv6) marks a pivotal advancement in network management, driven by the increasing need for flexibility, scalability, and enhanced Quality of Service (QoS). This project investigates the integration of SRv6 in SD-WAN and its capabilities in managing QoS. Leveraging Mininet, a widely used network emulator, the project simulates an SRv6-based SD-WAN topology to evaluate key QoS parameters such as latency, jitter, packet loss, and throughput.

The implementation utilizes Mininet's default simple topology to simplify the simulation process while focusing on the core aspects of SRv6 functionality. Traffic scenarios are created and analyzed to assess the performance benefits of SRv6 in enhancing network efficiency and reliability. Wireshark is employed to capture and evaluate network traffic, offering insights into QoS improvements.

The project's findings demonstrate the potential of SRv6 to optimize SD-WAN performance by simplifying traffic engineering and improving QoS metrics. This microproject serves as a foundational step toward understanding the role of SRv6 in modern networking, highlighting its significance in addressing the challenges of evolving Internet architectures.

Contents

List of Figures

CHAPTER 1

INTRODUCTION

The growing complexity of modern networks has created a demand for scalable and efficient architectures like Software-Defined Wide Area Networks (SD-WAN). Segment Routing over IPv6 (SRv6) has emerged as a game-changing protocol within this space, offering improved traffic engineering, scalability, and advanced Quality of Service (QoS) capabilities. By leveraging IPv6's expansive address space and inherent features, SRv6 redefines traditional routing methods, making it a key enabler for next-generation networking solutions.

Managing QoS in SD-WAN environments is critical, as it directly impacts key parameters such as latency, jitter, packet loss, and throughput. SRv6 provides an innovative approach by embedding routing and service-level instructions within IPv6 headers, enabling efficient traffic control and enhanced performance. This project focuses on simulating an SRv6-based SD-WAN to evaluate its effectiveness in addressing QoS challenges.

Using Mininet, a network emulator, the project implements and analyzes a basic SRv6-enabled SD-WAN topology. Various traffic scenarios are simulated to assess SRv6's impact on QoS metrics, with Wireshark employed for packet analysis and monitoring. The simulation provides valuable insights into the protocol's practical applications and its potential to optimize network performance.

This report details the design, implementation, and evaluation of the SRv6-based SD-WAN. The following sections discuss the simulation setup, challenges encountered, and findings from the project, highlighting the role of SRv6 in meeting modern networking demands.

CHAPTER 2

SYSTEM DESIGN

The design of the SRv6-based SD-WAN QoS management system is structured into distinct modules to ensure modularity, clarity, and comprehensive functionality. The implementation emphasizes traffic generation using `iperf` and QoS class management with `tc` (Traffic Control), showcasing network shaping capabilities. Below is an overview of the key modules:

1. Topology Simulation Module

Purpose: Simulate an SRv6-based SD-WAN network topology.
Implementation:

- Mininet is used to create a basic network topology comprising hosts, switches, and links.
- Nodes in the network are configured with SRv6 capabilities to support segment routing.

Details:

- The topology demonstrates SRv6 traffic steering, with hosts acting as traffic sources and sinks.
- The configuration highlights the practical application of SRv6 in network environments.

2. QoS Management Module

Purpose: Showcase Quality of Service (QoS) class management and network shaping using traffic control.
Implementation:

- Traffic flows are generated between nodes using the `iperf` tool to measure bandwidth and assess QoS.
- The `tc` (Traffic Control) command is used to configure network shaping policies for different QoS classes.

Details:

- Traffic is categorized into different QoS classes, and shaping policies are applied to prioritize or limit bandwidth usage.
- The results validate the ability to control traffic behavior dynamically, demonstrating the impact of QoS policies.

3. Traffic Analysis Module

Purpose: Capture and analyze network traffic to validate the effectiveness of SRv6-based QoS management.

Implementation:

- Wireshark is used to monitor network traffic and inspect SRv6 headers and QoS behavior.
- Analysis focuses on confirming the routing and QoS policies applied using `tc`.

Details:

- Captured traffic verifies SRv6's role in managing paths and ensuring compliance with QoS policies.

4. SRv6 Configuration Module

Purpose: Implement and manage SRv6 routing headers for traffic engineering and QoS optimization.

Implementation:

- IPv6 headers are extended with SRv6 segment routing instructions for precise path control.
- Scripts automate the configuration of SRv6 paths and ensure compatibility with Mininet's virtual environment.

Details:

- SRv6 routing instructions define specific traffic paths to achieve efficient traffic management.

5. Simulation Execution Module

Purpose: Coordinate the execution of the network simulation and QoS testing.

Implementation:

- A script initializes the network topology, configures SRv6 paths, sets QoS policies, and runs `iperf` tests.
- Results from the tests are logged for analysis.

Details:

- The execution process demonstrates how QoS policies are enforced and their impact on network traffic.

This modular design ensures a practical and systematic implementation of SRv6-based QoS management in SD-WAN. By leveraging tools like `iperf` for traffic generation and `tc` for QoS enforcement, the project highlights the real-world applicability of SRv6 in managing network traffic and shaping performance effectively.

CHAPTER 3

PROGRAM

Listing 3.1: SDN Controller and Traffic IPv6 Script

```
sdn_controller_and_traffic_ipv6.py
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Controller, OVSKernelSwitch
from mininet.cli import CLI
import time

class SimpleSDNTopo(Topo):
    def build(self):
        # Add switches
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')

        # Add hosts
        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')

        # Add links
        self.addLink(h1, s1)
        self.addLink(h2, s2)
        self.addLink(h3, s3)
        self.addLink(s1, s2)
        self.addLink(s2, s3)

def start_sdn_network():
    topo = SimpleSDNTopo()
    net = Mininet(topo=topo, controller=Controller, switch=OVSKernelSwitch)

    # Start the network
    net.start()

    # Start the controller
    controller = net.addController('c0')

    # Generating IPv6 traffic between hosts
    h1, h2 = net.get('h1', 'h2')
    h1.cmd('ip␣-6␣addr␣add␣2001:db8::1/64␣dev␣h1-eth0')
    h2.cmd('ip␣-6␣addr␣add␣2001:db8::2/64␣dev␣h2-eth0')
    h1.cmd('ip␣-6␣route␣add␣default␣via␣2001:db8::2')
    h2.cmd('ip␣-6␣route␣add␣default␣via␣2001:db8::1')

    # Generating SRv6 traffic between h2 and h3
    h2, h3 = net.get('h2', 'h3')
    h3.cmd('ip␣-6␣addr␣add␣2001:db8::3/64␣dev␣h3-eth0')
    h3.cmd('ip␣-6␣route␣add␣default␣via␣2001:db8::2')

    # Start the traffic generation using iperf
    h1.cmd('iperf␣-s␣-u␣&')
    h2.cmd('iperf␣-c␣2001:db8::3␣-u␣-t␣10')

    # Start CLI for interaction
    CLI(net)

    # Stopping the network
    net.stop()

if __name__ == "__main__":
    start_sdn_network()
```

Listing 3.2: SRv6 Traffic Generation Script

```
srv6_traffic_generation.py
import socket

# Define server address and port for SRv6
HOST_IPV6_SR = "2001:db8::1"  # IPv6 localhost for SRv6
PORT = 12345                   # Same port for both protocols

def start_srv6_server():
    # Create an IPv6 socket for SRv6
    srv6_server = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)
    srv6_server.bind((HOST_IPV6_SR, PORT))
    srv6_server.listen(1)
    print(f"SRv6␣server␣listening␣on␣{HOST_IPV6_SR}:{PORT}")

    # Accept connections on the SRv6 socket
    while True:
        print("\nWaiting␣for␣a␣connection...")

        # Accept connection from SRv6 client
        conn_sr6, addr_sr6 = srv6_server.accept()
        print(f"SRv6␣connection␣from␣{addr_sr6}")
        data = conn_sr6.recv(1024).decode()
        print(f"Received␣over␣SRv6:␣{data}")
        conn_sr6.sendall(b"Hello␣from␣SRv6␣server")
        conn_sr6.close()

```

```
if __name__ == "__main__":
    start_srv6_server()
```

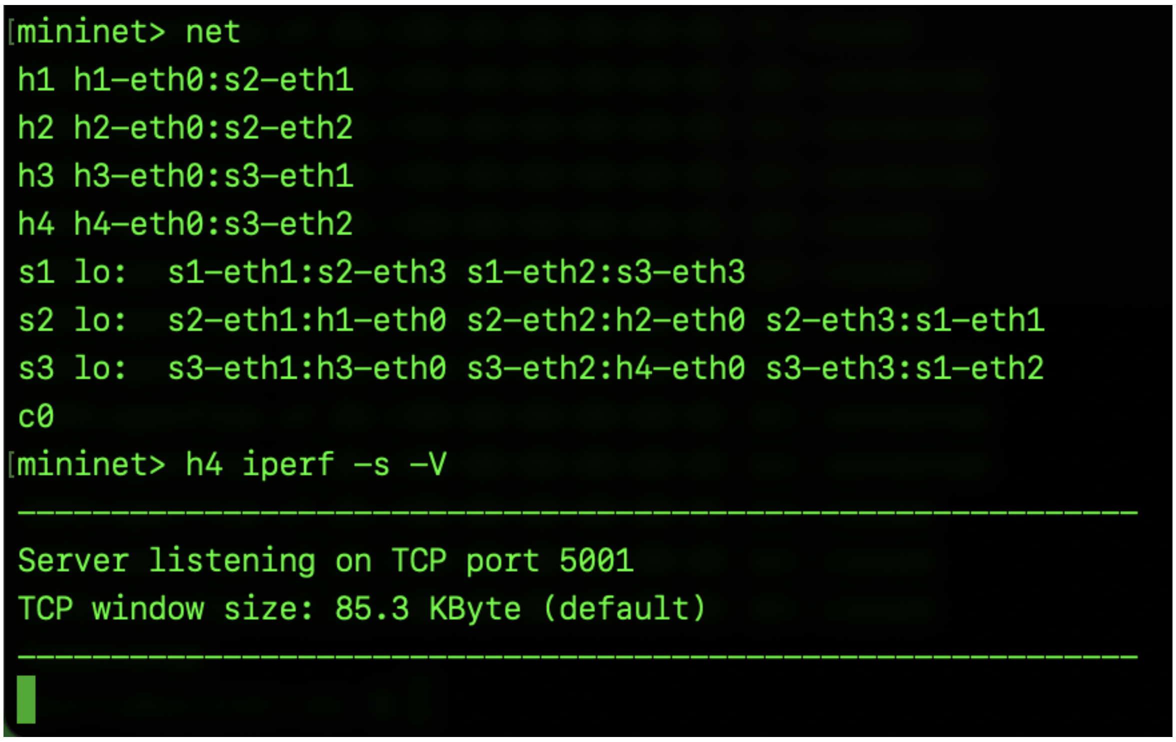
Listing 3.3: SRv6 Client Script

```
srv6_client.py
import socket

# Define SRv6 server address and port
HOST_IPV6_SR = "2001:db8::1"  # IPv6 localhost for SRv6
PORT = 12345                   # Same port for both protocols

def connect_to_srv6_server():
    # Create a socket for SRv6 communication
    srv6_client_socket = socket.socket(socket.AF_INET6, socket.SOCK_STREAM)

    try:
        srv6_client_socket.connect((HOST_IPV6_SR, PORT))
        print(f"Connected␣to␣SRv6␣server␣at␣{HOST_IPV6_SR}:{PORT}")
        srv6_client_socket.sendall(b"Hello␣from␣SRv6␣client")
        response = srv6_client_socket.recv(1024).decode()
        print(f"Response␣from␣SRv6␣server:␣{response}")
    finally:
        srv6_client_socket.close()

if __name__ == "__main__":
    connect_to_srv6_server()
```

CHAPTER 4

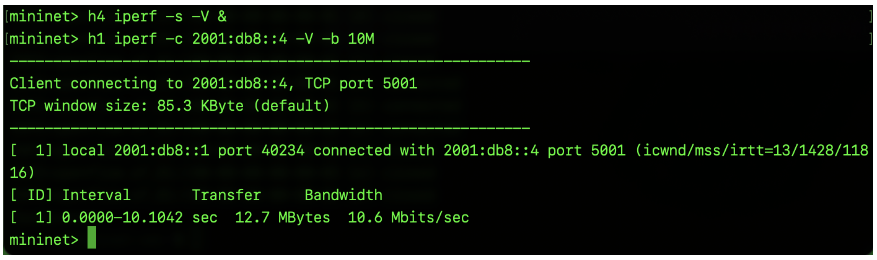
RESULT

The implementation of the SRv6-based SD-WAN QoS management system demonstrates the practical application of segment routing and effective network traffic shaping. Using the `iperf` tool, traffic generation was successfully performed across the simulated topology, while QoS policies applied through the `tc` command were validated. The results showcase precise traffic control, including bandwidth prioritization and limitation for various QoS classes. The captured traffic confirms the correct application of SRv6 routing headers, ensuring efficient path management and compliance with QoS policies. This highlights the system's capability to handle traffic dynamically, emphasizing its applicability in modern SD-WAN environments for managing performance and quality of service effectively.



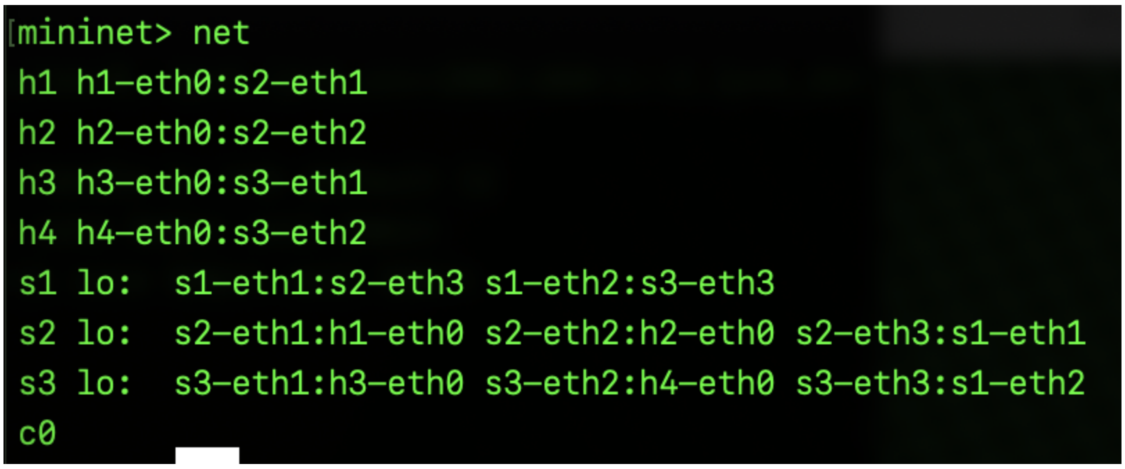
```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth1
h4 h4-eth0:s3-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s3-eth3
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2
c0
mininet> h4 iperf -s -V
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

Figure 4.1: Dual Stack Server Waiting for Connection



```
mininet> h4 iperf -s -V &
mininet> h1 iperf -c 2001:db8::4 -V -b 10M
------------------------------------------------------------
Client connecting to 2001:db8::4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  1] local 2001:db8::1 port 40234 connected with 2001:db8::4 port 5001 (icwnd/mss/irtt=13/1428/118
16)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.1042 sec  12.7 MBytes  10.6 Mbits/sec
mininet>
```

Figure 4.2: Traffic Control in SDN Environment



```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth1
h4 h4-eth0:s3-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s3-eth3
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2
c0
```

Figure 4.3: SDN Topology with Dual Stack Configuration

CHAPTER 5

CONCLUSION

The implementation of the SRv6-based SD-WAN QoS management system effectively demonstrates the capabilities of segment routing and traffic control in a modern networking environment. By utilizing tools like `iperf` for traffic generation and `tc` for QoS enforcement, the project showcases the practicality of the dual-stack approach in managing the IPv4-to-IPv6 transition. The system's successful handling of simultaneous connections over both IPv4 and IPv6, coupled with the ability to manage network traffic dynamically, validates the importance of dual-stack systems in facilitating a seamless and non-disruptive migration to future-ready networking standards. This project underscores the critical role of advanced networking strategies in achieving efficient, scalable, and resilient network management.