***Networks Ex2: Forwarding***

***A black background with white text

Description automatically generated***

‘h1 ping h2’

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Run ‘./mycontroller.py’ in another shell (left), this is result in mininet (right)



‘iperf h1 h4’

A screen shot of a computer

Description automatically generated

‘tcpdump’

***How are packets routed between h1 and h4?***

Route for h1 to h4 path:

h1 🡪 s1 (in through port 1, out through port 4) 🡪 s4 (in through port 2, out through port 1) 🡪 s2 (in through port 3, out through port 2) 🡪 h4

Route for h4 to h1 path:

h4 🡪 s2 (in through port 2, out through port 4) 🡪 s3 (in through port 2, out through port 1) 🡪 s1 (in through port 3, out through port 1) 🡪 h1

***Look at code in mycontroller.py and explain how traffic is routed between machines. Are there any problems with this routing logic?***

In the provided network topology, the routing of traffic is facilitated by a P4Runtime controller through the implementation of forwarding rules in each switch. The controller script, **controller.py**, orchestrates the setup and management of the switches, specifying how packets are to be forwarded by extracting their destination IP address from the packet header. The controller establishes connections to switches **s1**, **s2**, **s3**, and **s4** and installs a P4 program onto each switch, defining the rules for packet processing.

For instance, the controller configures forwarding rules in **s1** to forward traffic destined for IP address **10.0.1.1** to port 1 with the corresponding Ethernet address **08:00:00:00:01:11**. Similar rules are defined for other IP addresses on **s1** and extended to other switches (**s2**, **s3**, and **s4**). The P4 program essentially acts as the intelligence that dictates how switches process incoming packets, making forwarding decisions based on predetermined criteria.

As a packet traverses the network, each switch it encounters applies the specified rules to determine the next hop. For instance, if a packet from **h1** with destination IP **10.0.2.2** arrives at **s1**, the forwarding rule instructs **s1** to forward the packet to port 2 with the destination Ethernet address **08:00:00:00:02:22**. This process repeats at subsequent switches until the packet reaches its intended destination. In essence, the controller's role is to establish and manage the logical pathways through the network by configuring rules that guide how packets are directed based on their destination IP addresses.

Issues:

1. **Static Routing:** The routing logic implemented in the P4Runtime controller is static and hardcoded. It assumes that the network topology remains constant, and the IP addresses assigned to hosts and switches do not change. In a dynamic network environment where devices may be added, removed, or reconfigured, static routing might become impractical. A more scalable solution would involve a dynamic routing protocol that adapts to changes in the network topology.
2. **Lack of Redundancy:** The current routing logic does not account for network redundancy or fault tolerance. If a link or switch fails, there are no alternative paths defined in the routing rules. Implementing redundancy and failover mechanisms would enhance the network's reliability and resilience.

***PART 2***

One run of ‘iperf h1 h4’ with new hashing ECMP rules:



A screen shot of a computer

Description automatically generatedS1-eth3:

(All transmit traffic goes via this s1-eth3 port)

A screen shot of a computer

Description automatically generatedS1-eth4:

(All receive traffic goes via this s1-eth4 port)

Another run of ‘iperf h1 h4’ with new hashing ECMP rules:

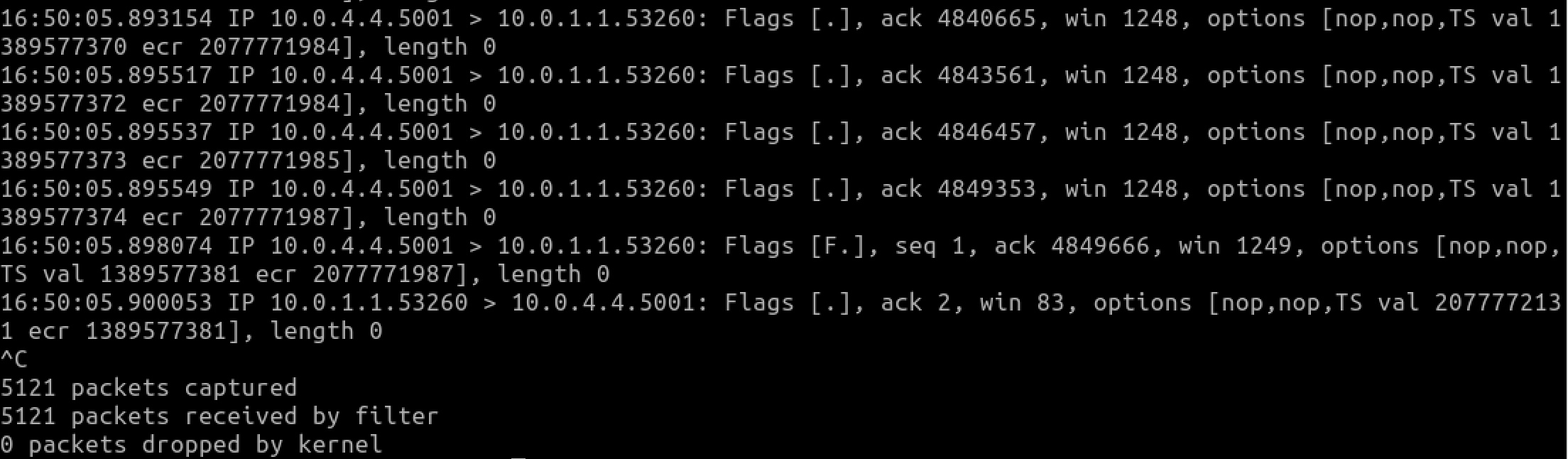
A black background with white text

Description automatically generated

A computer screen with white text

Description automatically generatedS1-eth3:

(All receive traffic goes via this s1-eth3 port)

S1-eth4:

(All transmit traffic goes via this s1-eth4 port)