CN Lab Part-II

Wireshark

- The world's most popular network protocol analyzer.

- Wireshark is a **network packet analyzer**. A network packet analyzer presents captured packet data in as much detail as possible.

Some purposes of Wireshark:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

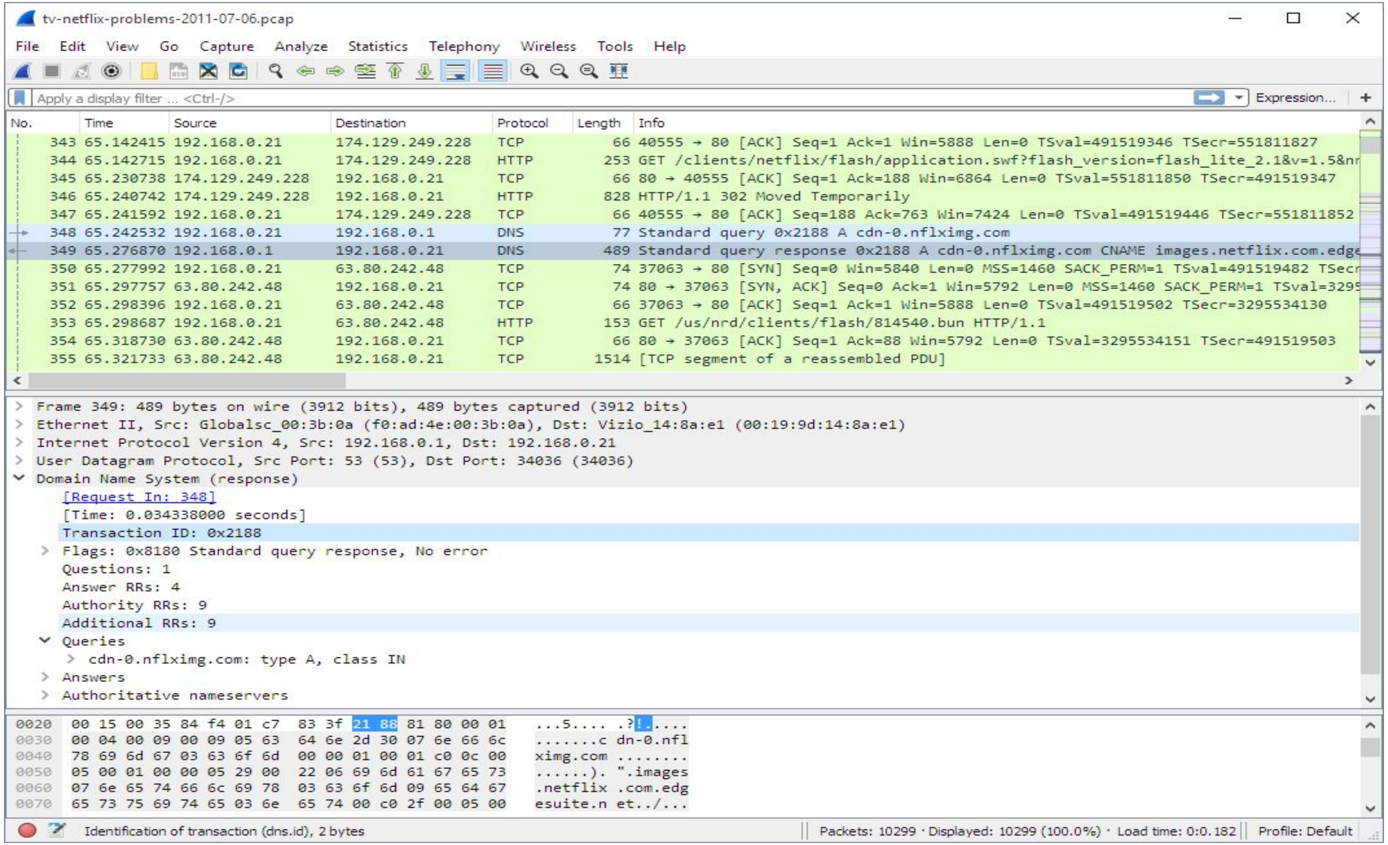
Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

Wireshark

Here are some things Wireshark does not provide:

- Wireshark isn’t an intrusion detection system.
- Wireshark will not manipulate things on the network, it will only “measure” things from it.

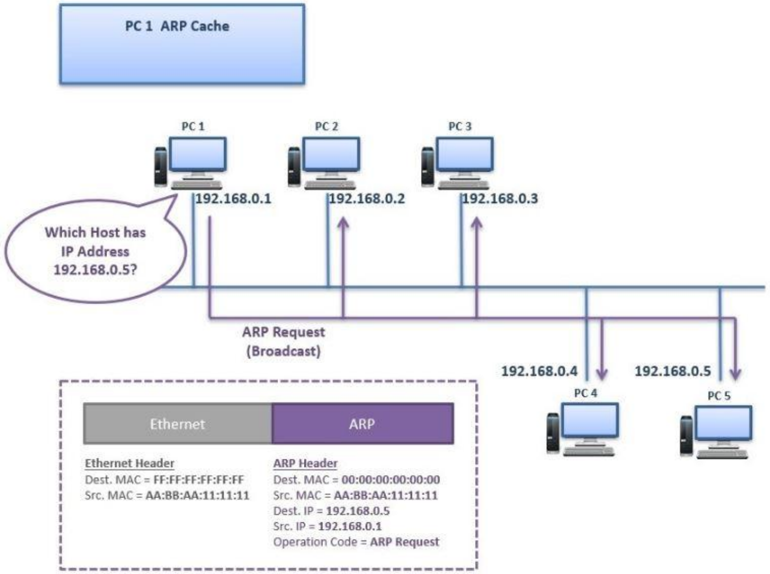
Wireshark



tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
343 65.142415 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344 65.142715 192.168.0.21 174.129.249.228 HTTP 253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
345 65.230738 174.129.249.228 192.168.0.21 TCP 66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
346 65.240742 174.129.249.228 192.168.0.21 HTTP 828 HTTP/1.1 302 Moved Temporarily
347 65.241592 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
348 65.242532 192.168.0.21 192.168.0.1 DNS 77 Standard query 0x2188 A cdn-0.nflximg.com
349 65.276870 192.168.0.1 192.168.0.21 DNS 489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
350 65.277992 192.168.0.21 63.80.242.48 TCP 74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
351 65.297757 63.80.242.48 192.168.0.21 TCP 74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
352 65.298396 192.168.0.21 63.80.242.48 TCP 66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
353 65.298687 192.168.0.21 63.80.242.48 HTTP 153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
354 65.318730 63.80.242.48 192.168.0.21 TCP 66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
355 65.321733 63.80.242.48 192.168.0.21 TCP 1514 [TCP segment of a reassembled PDU]
Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
Domain Name System (response)
[Request In: 348]
[Time: 0.034338000 seconds]
Transaction ID: 0x2188
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 4
Authority RRs: 9
Additional RRs: 9
Queries
cdn-0.nflximg.com: type A, class IN
Answers
Authoritative nameservers
0020 00 15 00 35 84 f4 01 c7 83 3f 21 88 81 80 00 01 ...5.... .?!.....
0030 00 04 00 09 00 09 05 63 64 6e 2d 30 07 6e 66 6c ....c dn-0.nfl
0040 78 69 6d 67 03 63 6f 6d 00 00 01 00 01 c0 0c 00 ximg.com ....
0050 05 00 01 00 00 05 29 00 22 06 69 6d 61 67 65 73 ......). ".images
0060 07 6e 65 74 66 6c 69 78 03 63 6f 6d 09 65 64 67 .netflix .com.edg
0070 65 73 75 69 74 65 03 6e 65 74 00 c0 2f 00 05 00 esuite.n et../...
Identification of transaction (dns.id), 2 bytes
Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182
Profile: Default

Address Resolution Protocol (ARP)

- ARP (Address Resolution Protocol) is a Layer 2 Protocol.
- Layer 2 uses Physical addresses (MAC addresses) and Layer 3 uses Logical addresses (IP Addresses) for the communication.
- ARP Protocol is used to discover the MAC Address of a node associated with a given IPv4 Address.



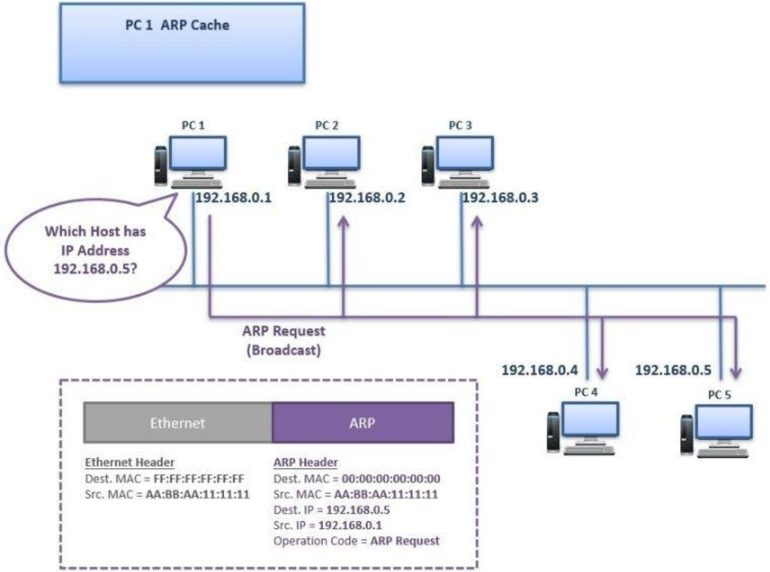
Address Resolution Protocol (ARP)

PC 1 sends an “ARP Request” Message to the network as broadcast. This ARP Request is sent to all the nodes in the network.

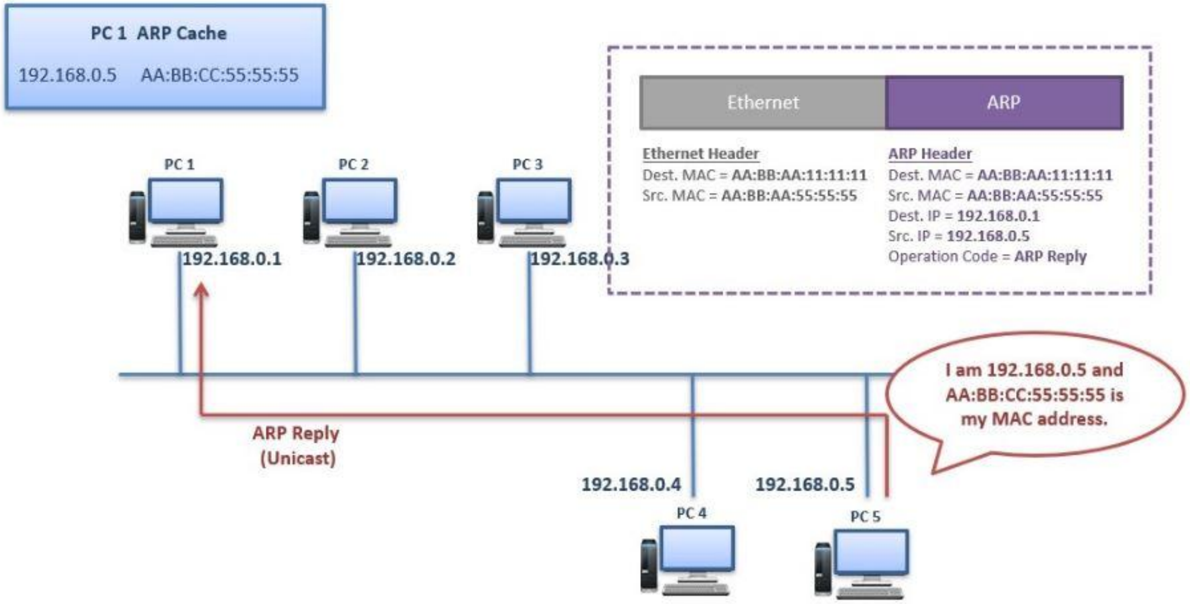
“Which Host has IP Address 192.168.0.5?”

This ARP Request Message consists of source and destination IP, source MAC address and operation code “Request”. Destination MAC is written as 00:00:00:00:00:00:00 means it is requested.

In the Layer 2 header of this message, the destination MAC is FF:FF:FF:FF:FF:FF. This is the broadcast MAC address.



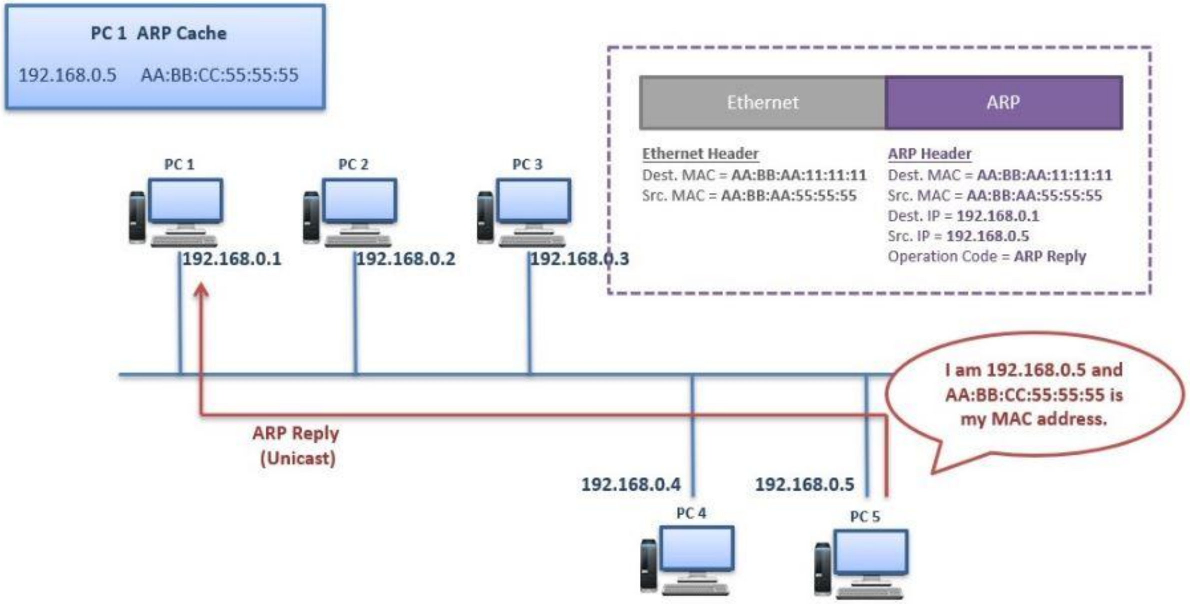
Address Resolution Protocol (ARP)



PC 5 replies to this ARP Request Message with an “ARP Reply” Message. PC 5 sends this ARP Reply Message directly to PC 1 as unicast message.

“I am 192.168.0.5 and this AA:BB:CC:55:55:55 is my MAC address.”

Address Resolution Protocol (ARP)



Test on Wireshark

IPERF Tool (Internet Performance)

- ❑ It is a commonly used tool in Mininet to test network performance, including bandwidth, latency, and jitter between hosts.
- ❑ Intall iperf using command: **sudo apt-get install iperf iperf3**

1. Set Up Mininet Topology

```
sudo mn --topo=single,2 --controller=none
```

2. Start iperf Server on One Host

```
mininet> h1 iperf -s
```

3. Start iperf client on another host

```
mininet> h2 iperf -c 10.0.0.1
```

```
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 5201 connected with 10.0.0.1 port 5201
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  1.09 GBytes  933 Mbits/sec
```

Sample output

IPERF Tool (Internet Performance)

- Bidirectional Test:

```
mininet> h2 iperf -c 10.0.0.1 -d
```

- UDP Test:

```
mininet> h2 iperf -c 10.0.0.1 -u
```

- Adjusting the Bandwidth for UDP Tests

```
mininet> h2 iperf -c 10.0.0.1 -u -b 10M
```

IPERF Tool (Internet Performance)

- Parallel connection:

```
mininet> h2 iperf -c 10.0.0.1 -P 5
```

Parallel connection output:

```
Connecting to host 10.0.0.1, port 5201
[  4] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 5201
[  5] local 10.0.0.2 port 5002 connected with 10.0.0.1 port 5201
[  6] local 10.0.0.2 port 5003 connected with 10.0.0.1 port 5201
[ ID] Interval       Transfer     Bandwidth
[  4] 0.0-10.0 sec  50 MBytes    42 Mbits/sec
[  5] 0.0-10.0 sec  50 MBytes    42 Mbits/sec
[  6] 0.0-10.0 sec  50 MBytes    42 Mbits/sec
[SUM] 0.0-10.0 sec  150 MBytes   126 Mbits/sec
```

ARP Analysis and Address Resolution using Wireshark

Objective: Capture ARP traffic to understand how IP addresses are mapped to MAC addresses on a local network.

1. **Set Up Mininet Topology:** Launch Mininet with a simple topology, such as one switch with two hosts:

   sudo mn --topo=single,2 --controller=none

2. **Start Wireshark Capture:** Start capturing on your LAN interface

3. **Generate ARP Traffic:** Use a command like ping to a local device to generate ARP requests.

4. **Filter ARP Packets:** Use the filter arp to view ARP traffic

5. **Analyze ARP Requests and Replies:** Examine how devices request MAC addresses using IP addresses and respond

TCP Handshake Analysis using Wireshark

Objective: Capture and analyze the TCP three-way handshake and connection termination.

1. **Set Up Mininet Topology:** Launch Mininet with a simple topology, such as one switch with two hosts:

   sudo mn --topo=single,2 --controller=none

2. **Start Wireshark Capture:** Start capturing on your LAN interface

3. **Generate TCP Traffic:** Use iperf to generate TCP traffic

4. **Filter TCP Packets:** Use the filter tcp to view TCP traffic

5. **Identify the Three-Way Handshake:** Look for SYN, SYN-ACK, and ACK packets.

Observe how the handshake establishes a connection and the termination sequence with FIN and ACK

Analyzing Packet Loss and Retransmissions using Wireshark

Objective: Detect and analyze packet loss and TCP retransmissions.

1. **Set Up Mininet Topology:** Launch Mininet with a simple topology, such as one switch with two hosts:

sudo mn --topo=single,2 --controller=none

2. **Start Wireshark Capture:** Start capturing on your LAN interface

3. **Generate TCP Traffic:** Use iperf to generate TCP traffic

4. **Identify Packet Loss and Retransmissions:** Use filters like **tcp.analysis.retransmission** and **tcp.analysis.lost_segment**

5. **Analyze Impact on Performance:** Observe how retransmissions affect throughput and latency.

Check the ARP Table in Mininet

Objective: The ARP table maps IP addresses to MAC addresses on the network, which is essential for communication between devices on the same local network.

1. **Set Up Mininet Topology:** Launch Mininet with a simple topology, such as one switch with two hosts:

   sudo mn --topo=single,3 --controller=none

2. **Generate ARP Entries:** Ping between Hosts to generate ARP entries

   mininet> h1 ping h2 –c 1
   mininet> h1 ping h3 –c 1

4. **Check the ARP Table on a Host:** Access a Host's CLI and view the ARP table using the following commands:

   mininet> h1 arp -n

   Another method:

   mininet> h1 ip neigh

5. **Clear the ARP Table:** mininet> h1 ip neigh flush all

Traffic Flow

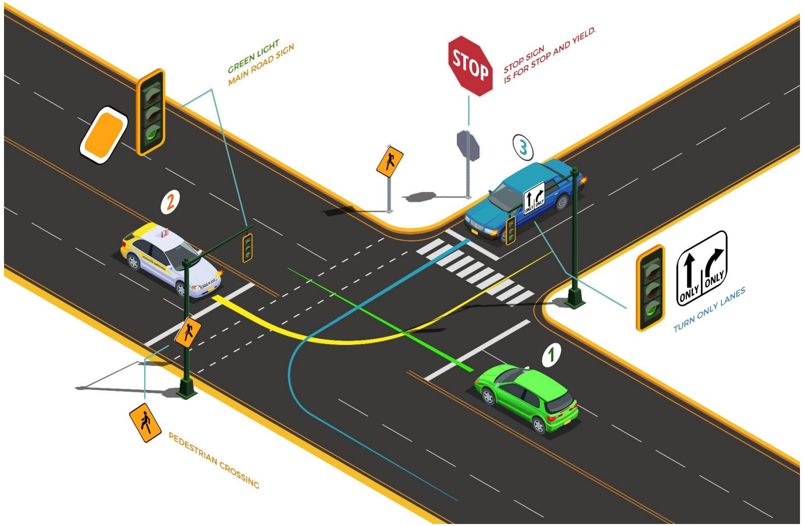


Fig 1. Traffic flow

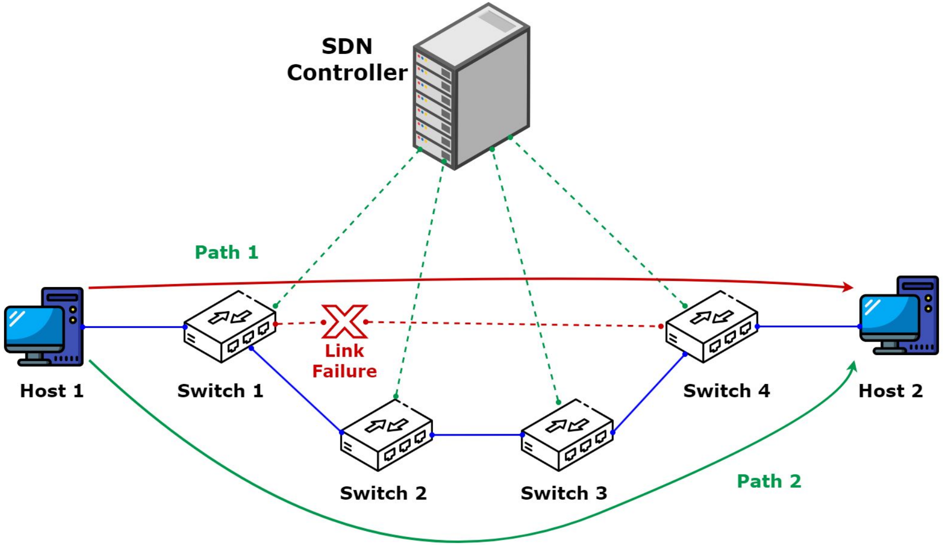


Fig 2. SDN Traffic flow

Software-Defined Networking (SDN)

Definition: Software-Defined Network provides a novel paradigm architecture of the network by decoupling the control plane from the data plane.

- SDN offers programmability, adjustable, and dynamic configuration of the networking devices.
- Unlike a traditional network, the SDN paradigm introduces centralized control structure which dynamically configures the forwarding devices.
- The controller has the responsibility for installing and manipulating the flow rules in the switches of the data plane.

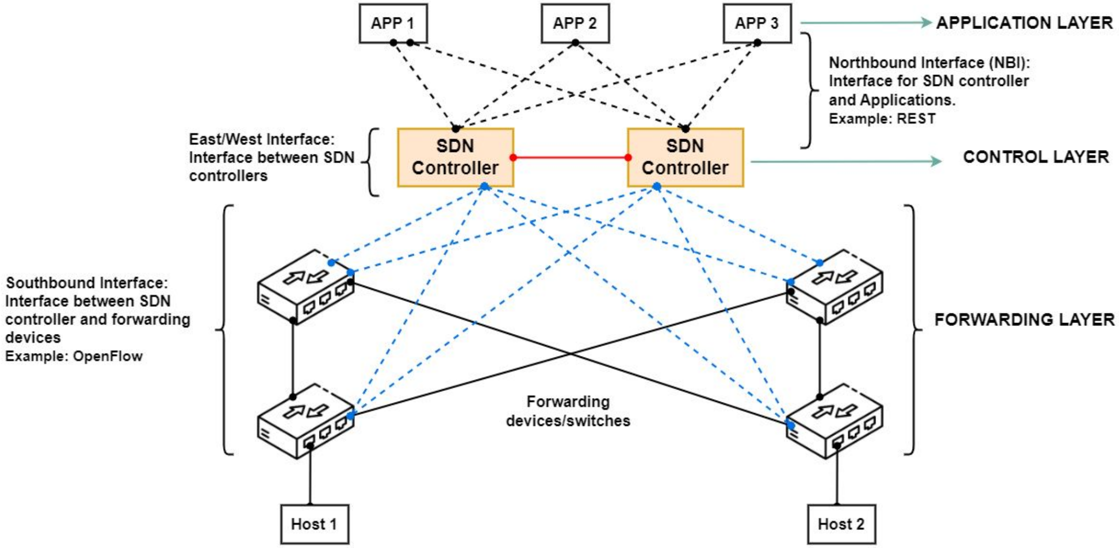
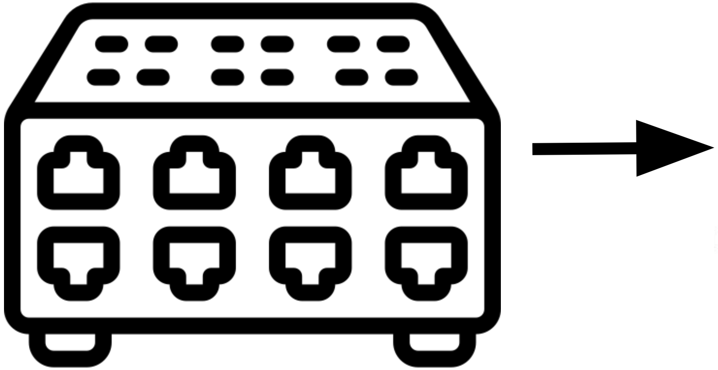
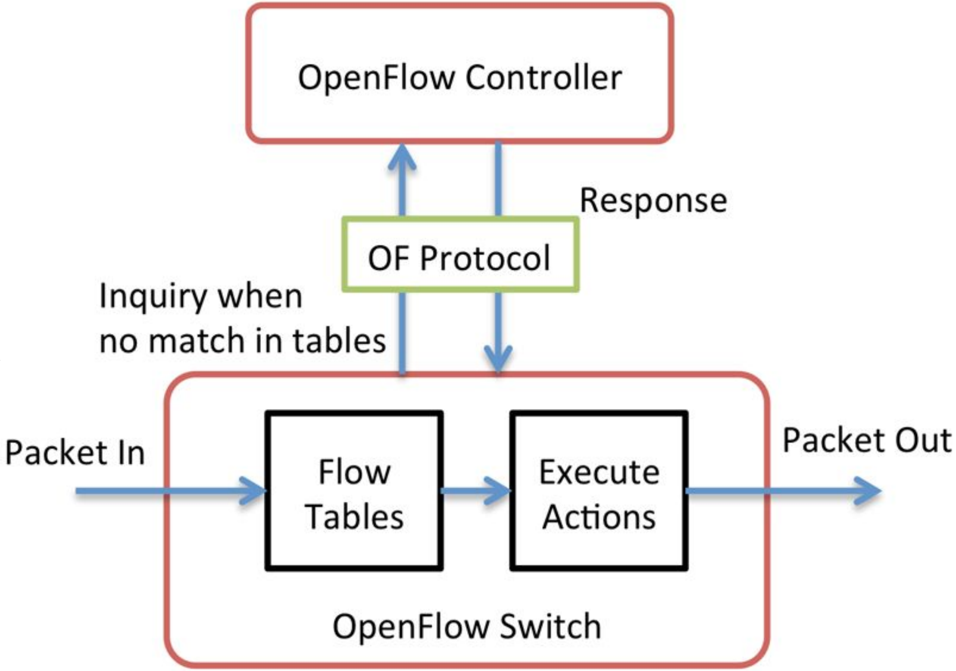


Fig 1. SDN architecture

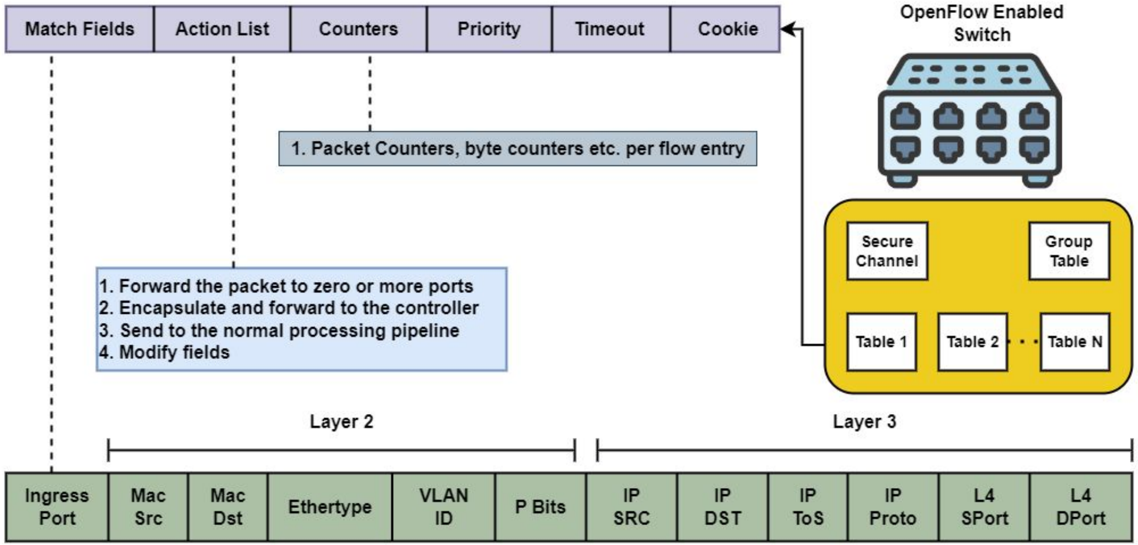
OpenFlow Protocol

Definition: A standard protocol to communicate between the switches and the software-based controller in an SDN network.





OpenFlow Structure



- Each forwarding device maintains one or more flow tables, which contain the rules that dictate how incoming packets should be handled.
- When a packet arrives at an OpenFlow-enabled switch, the switch matches the packet's header fields against its flow table entries.
- If a match is found, the specified action (forward to a port, modify the packet, drop the packet) is executed. If no match is found, the packet can be sent to the controller for further processing.

Managing/viewing Manual OpenFlow Rules on Switches

1. View All Flows on the Switch: **sudo ovs-ofctl dump-flows s1**
2. Delete All Flows on the Switch: **sudo ovs-ofctl del-flows s1**

Adding Manual OpenFlow Rules on Switches

1. View the installed flow rules on the: **sudo ovs-ofctl dump-flows s1**

2. Set Up Mininet Topology: Launch Mininet with a simple topology, such as one switch with two hosts:

sudo mn --topo=single,2 --controller=none

3. Check the Switch Ports: To add flow rules, you need to know the port numbers connected to each host

sudo ovs-ofctl show s1

4. Add Flow Rules to the Switch:

```
sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2
sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1
```

5. Test the Configuration: **mininet> h1 ping -c 1 h2**

Adding Manual OpenFlow Rules on Switches

Task 1. Drop All Traffic (Deny Traffic by Default):

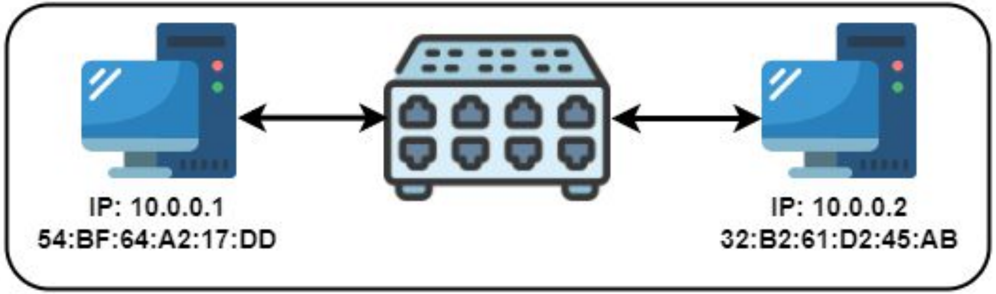
```
sudo ovs-ofctl add-flow s1 priority=0,actions=drop
```

Task 2. Forward ICMP (Ping) Traffic Only:

```
sudo ovs-ofctl add-flow s1 in_port=1,icmp,actions=output:2
sudo ovs-ofctl add-flow s1 in_port=2,icmp,actions=output:1
```

Task 3. Flood Traffic to All Ports Except Incoming Port:

```
sudo ovs-ofctl add-flow s1 in_port=1,actions=FLOOD
```



sudo ovs-ofctl add-flow s1 in_port=1,actions=output:2
sudo ovs-ofctl add-flow s1 in_port=2,actions=output:1

Step 1. Add rules
Step 2. Try ping command

Adding Manual OpenFlow Rules on Switches

Task 4. Layer 2 Switching Rules

- Forward Based on MAC Address: Forward packets with a specific destination MAC address to a specific port.

sudo ovs-ofctl add-flow s1 dl_dst=00:00:00:00:00:02,actions=output:2

Task 5. Layer 3 Routing Rules

- Forward Based on IP Address: Forward packets with a specific destination IP address to a particular port.

sudo ovs-ofctl add-flow s1 ip,nw_dst=10.0.0.2,actions=output:2

Task 6. Layer 3 Routing Rules

- Drop Packets from a Specific IP Address:

sudo ovs-ofctl add-flow s1 ip,nw_src=10.0.0.3,actions=drop

Adding Manual OpenFlow Rules on Switches

Task 7. Protocol-Based Forwarding Rules

- Forward ICMP (Ping) Traffic:

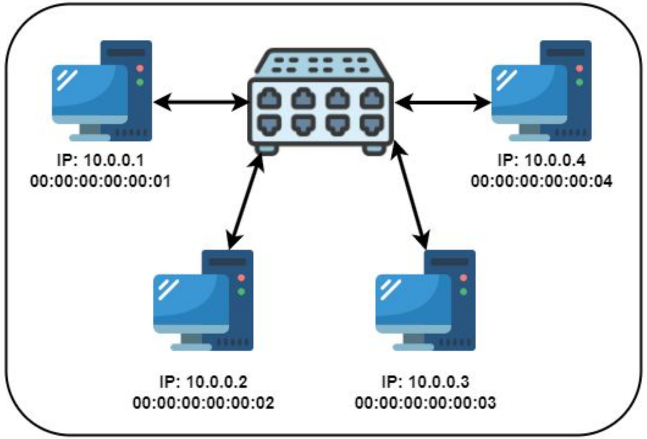
```
sudo ovs-ofctl add-flow s1 in_port=1,icmp,actions=output:2
sudo ovs-ofctl add-flow s1 in_port=2,icmp,actions=output:1
```

Task 8. Forward TCP Traffic on Port 80 (HTTP):

```
sudo ovs-ofctl add-flow s1 in_port=1,ip,nw_proto=6,tp_dst=80,actions=output:2
sudo ovs-ofctl add-flow s1 in_port=2,ip,nw_proto=6,tp_dst=80,actions=output:1
```

Assignment 5

- You are given the following network topology created in Mininet. The switch is configured without a controller, and you are required to manually add OpenFlow rules using **ovs-ofctl** commands on the switch (s1). The objective is to set up traffic rules based on the following requirements:



- ✔ **Objective a: H1** should be allowed to send ICMP (ping) traffic to **H2** and **H3**, but not to any other hosts.
- ✔ **Objective b:** No other types of traffic should be allowed from **H1**
- ✔ **Objective c: H2** is allowed to send TCP traffic to **H4**. ICMP traffic from **H2** to any host should be blocked.
- ✔ **Objective d: H3** should be able to communicate freely with **H4** but should be restricted from sending any traffic to **H1**
- ✔ **Objective e:** All traffic originating from **H4** should be blocked, making it unable to send packets to any other host

Adding Manual OpenFlow Rules on Switches

Task 9. VLAN Tagging and Modification

- Tag Incoming Traffic with VLAN ID 100:

```
sudo ovs-ofctl add-flow s1 in_port=1,actions=mod_vlan_vid:100,output:2
```

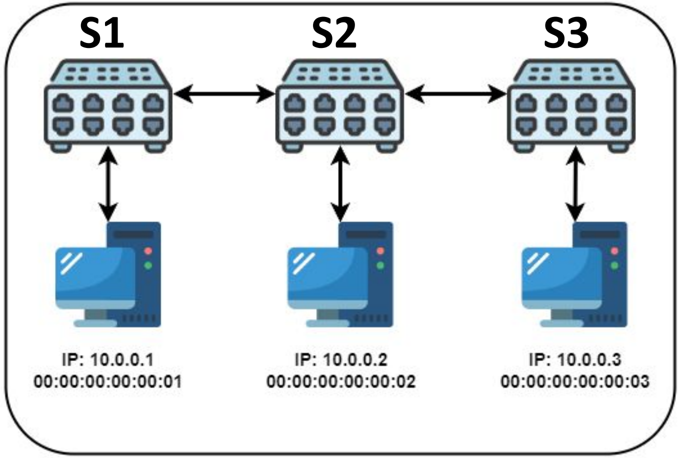
Task 10. Strip VLAN Tag and Forward:

```
sudo ovs-ofctl add-flow s1 dl_vlan=100,actions=strip_vlan,output:3
```

Assignment 6

- ❑ You are given the following network topology created in Mininet. The switch is configured without a controller, and you are required to manually add OpenFlow rules using **ovs-ofctl** commands on the switch (s1, s2, and s3). The objective is to set up traffic rules based on the following requirements:

- ✔ **Objective a: H1** should be allowed to send ICMP (ping) traffic to **H2** and **H3**, but not to any other hosts.
- ✔ **Objective b:** No other types of traffic should be allowed from **H1**
- ✔ **Objective c: H2** is allowed to send TCP traffic to **H4**. ICMP traffic from **H2** to any host should be blocked.



Assignment 7

❑ You are given the following network topology created in Mininet. The switch is configured without a controller, and you are required to manually add OpenFlow rules using **ovs-ofctl** commands on the switch (s1, s2, and s3). The objective is to set up traffic rules based on the following requirements:

✔ **Objective a: H1** should be allowed to send ICMP (ping) traffic to **H2** and **H3**, but not to any other hosts.

✔ **Objective b:** No other types of traffic should be allowed from **H1**

✔ **Objective c: H2** is allowed to send TCP traffic to **H4**. ICMP traffic from **H2** to any host should be blocked.