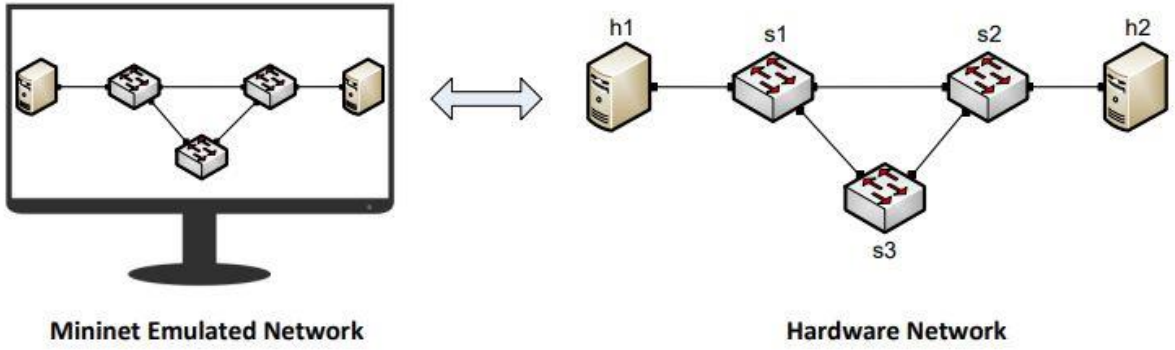
Mininet Emulation Tool

Mininet Emulation Tool

❑ Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software.



❑ Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network.

Mininet Emulation Tool

- Provides a simple and inexpensive network testbed for developing applications.

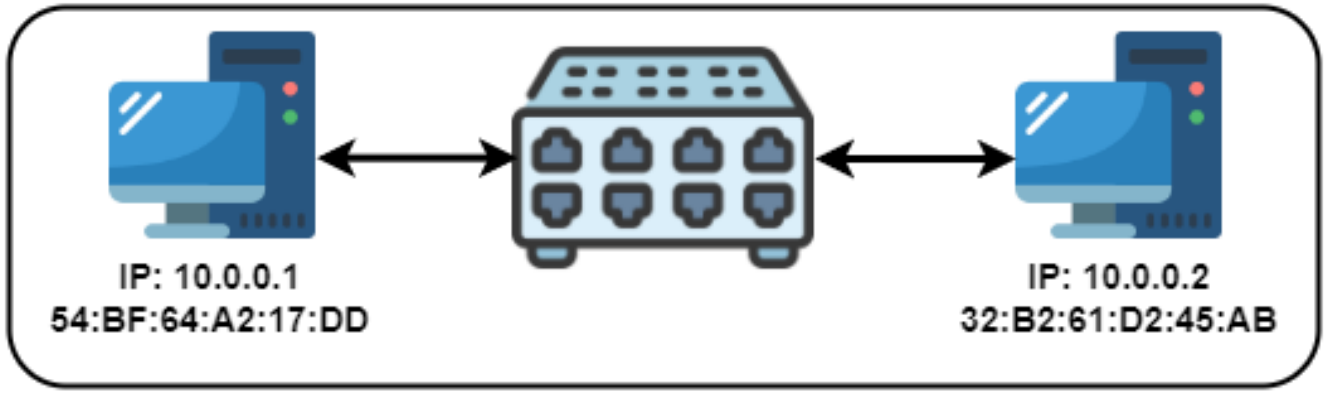
- Enables complex topology testing, without the need to wire up a physical network.

- Mininet provides an easy way to get correct system behaviour (and, to the extent supported by your hardware, performance) and to experiment with topologies.

Mininet Commands

sudo mn

This commands creates a single virtual openvswitch and two virtual hosts with default IP and mac configuration (random).



Type ***exit*** command to come out of the mininet.
Then, destroy any remaining components using ***sudo mn –c***

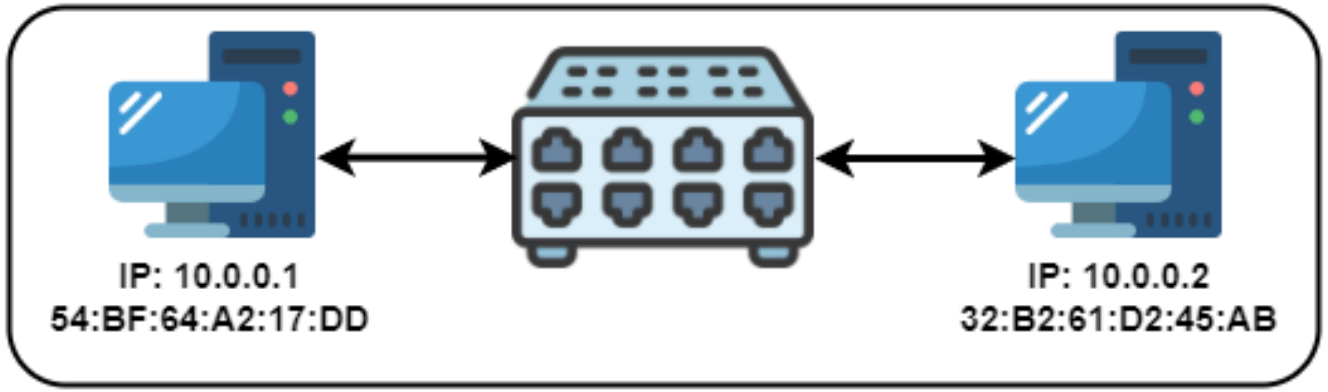
Mininet Commands

mininet> pingall
mininet> links
mininet> nodes
mininet> net
mininet> dump

mininet> h1 ifconfig

Link up/down Command

mininet> link s1 h1 down
mininet> link s1 h1 up



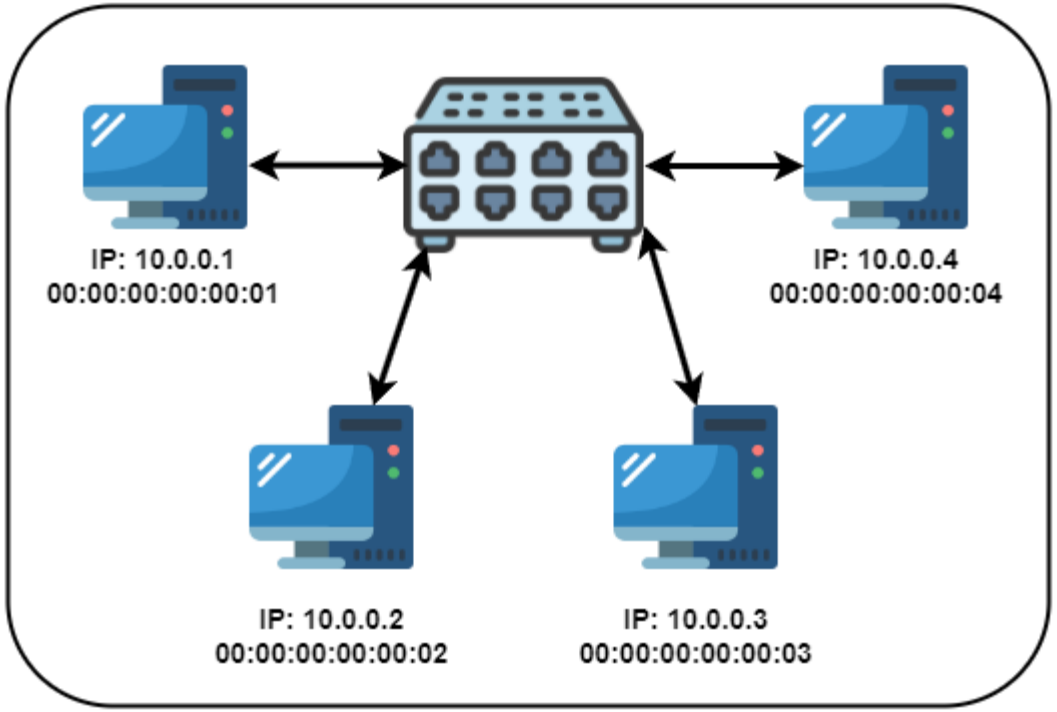
Mininet Commands

Different Topologies

sudo mn –topo single,4 --mac

This commands creates a single virtual openvswitch and four virtual hosts with default IP and mac configuration.

Here ***--mac*** is used to generate serial mac addresses for the host.



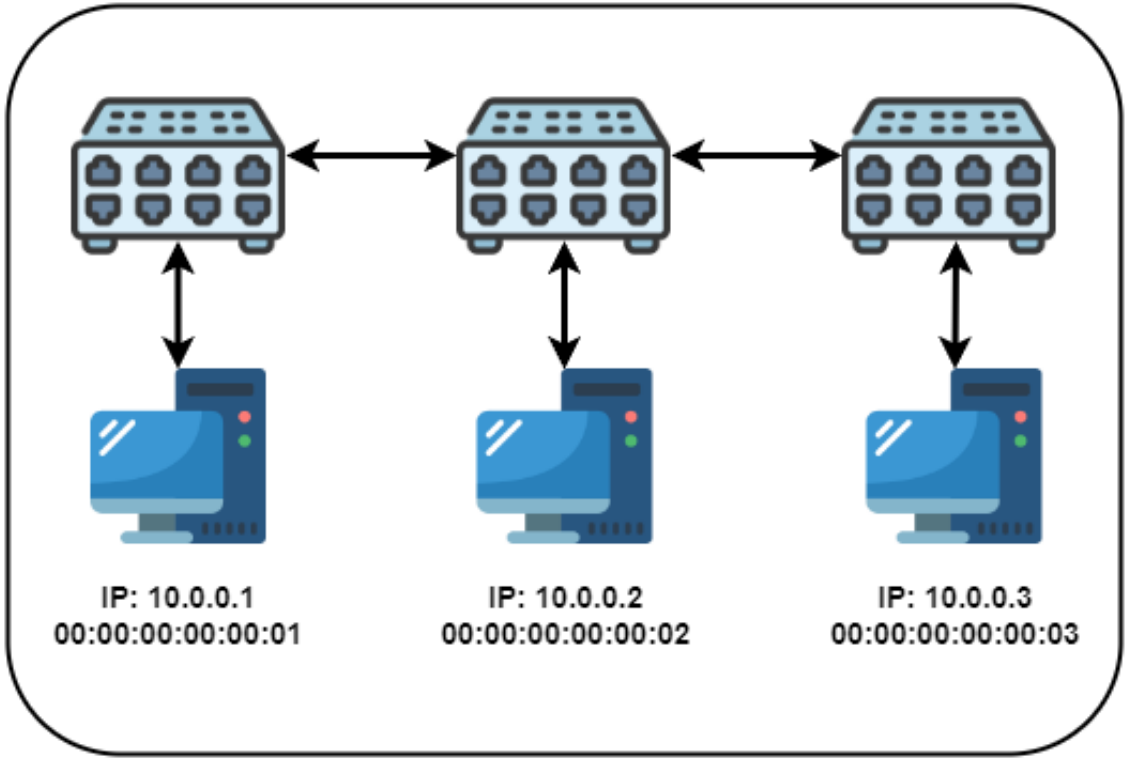
Mininet Commands

Linear Topology

sudo mn –topo linear,3 --mac

This commands creates a linear topology where 3 virtual switches are connected to each other.

Each switch is having one host connected to it.



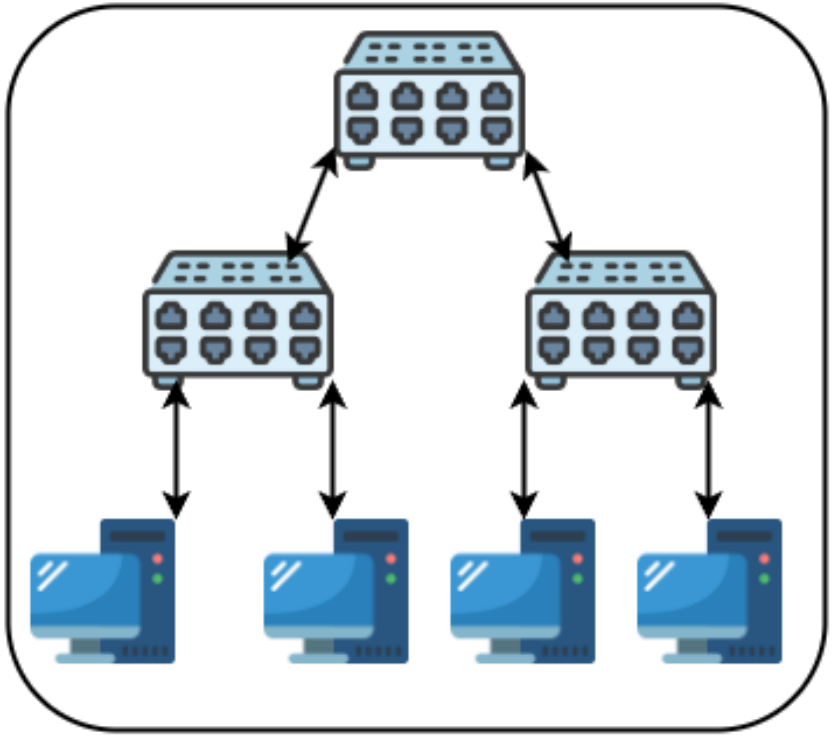
Mininet Commands

Tree Topology

sudo mn --topo tree,depth=2,fanout=2 --mac

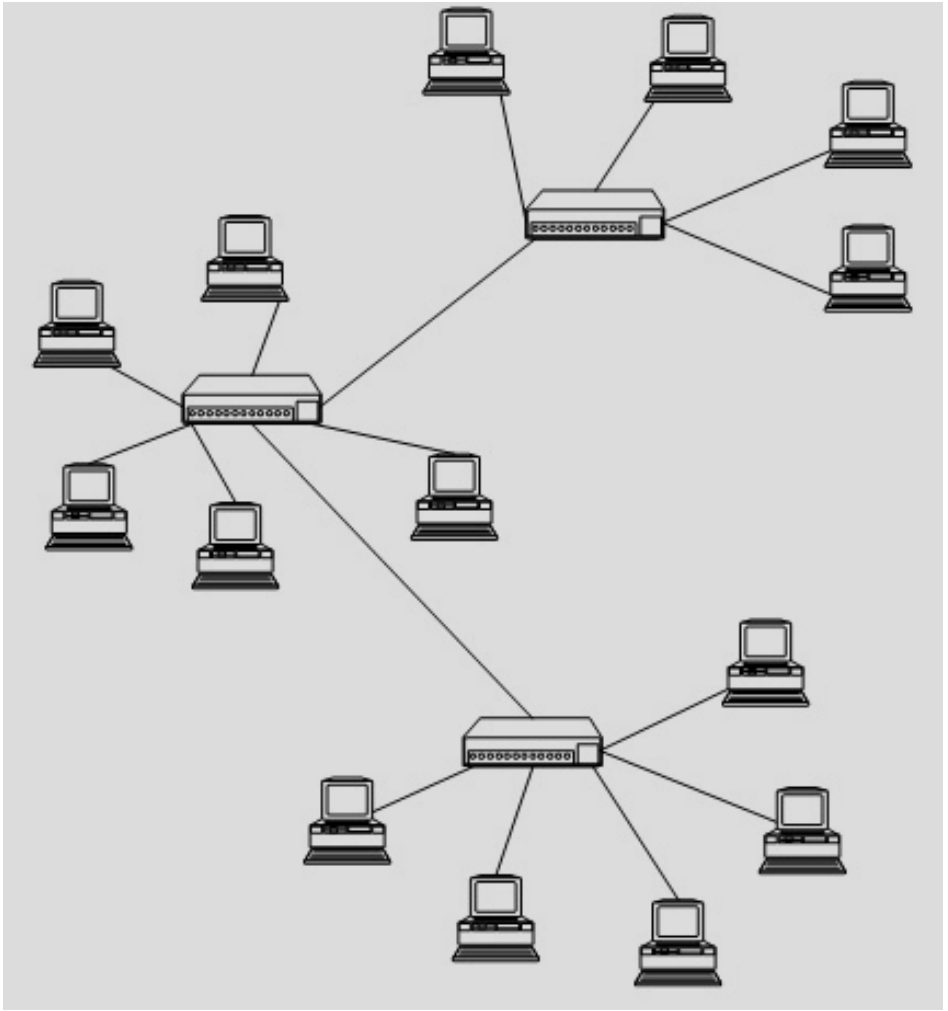
This commands creates a tree topology where 3 virtual switches are connected to each other in tree form.

The depth signifies the number of switches and fanout signifies the number of host on each leaf switches.



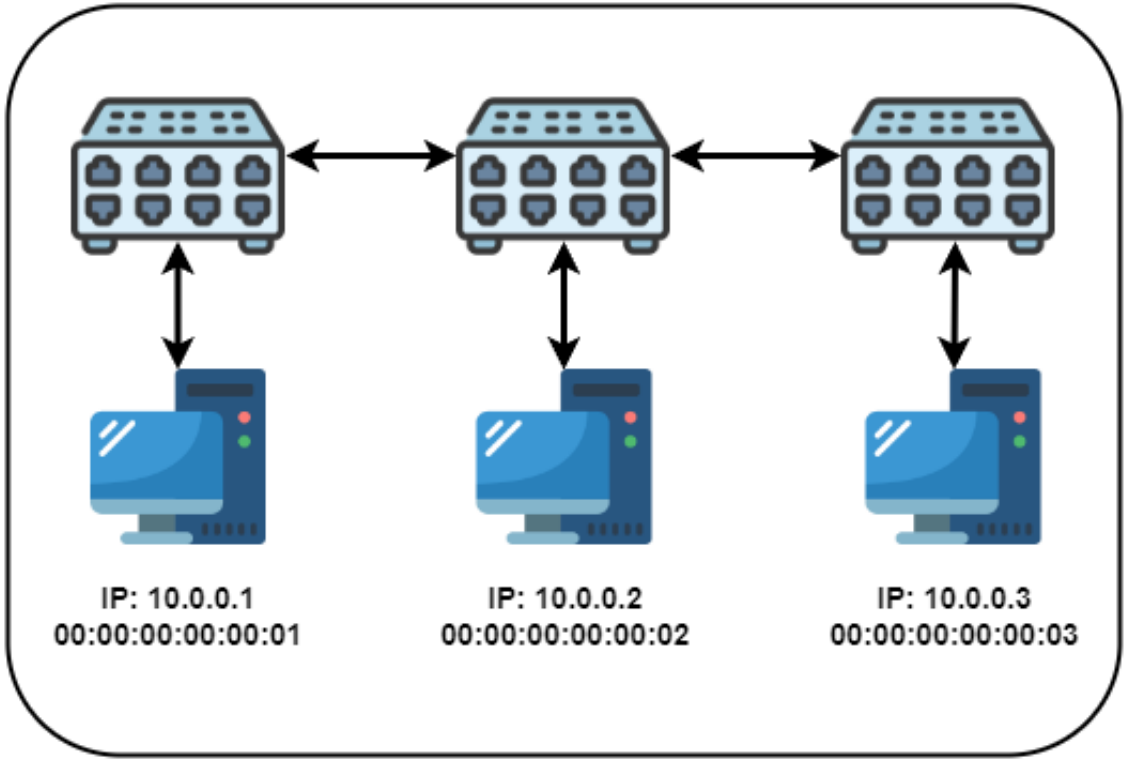
Custom Topologies

- The standard Mininet commands are limited to only few topologies.
- We can build a network topology using programming language such as python for a more complex network.
- The class for creating network components are defined in the python library as a package. Therefore, we can import those library and build the network topology according to our requirement.



Custom Topologies

```
from mininet.topo import Topo
class MyTopo( Topo ):
    def __init__( self ):
        Topo.__init__( self )

        # Add hosts and switches
        Host1 = self.addHost( 'h1' )
        Host2 = self.addHost( 'h2' )
        Host3 = self.addHost( 'h3’ )

        Switch1 = self.addSwitch('s1')
        Switch2 = self.addSwitch('s2’)
        Switch3 = self.addSwitch(‘s3')

        # Add links
        self.addLink( Host1, Switch1 )
        self.addLink( Host2, Switch1 )
        self.addLink( Host3, Switch2 )
        self.addLink( Switch1, Switch2 )
        self.addLink( Switch2, Switch3 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```



Run the command

sudo mn --custom custom_topo.py --topo=mytopo

Custom Topologies

Q: How can we add custom mac and IP address to the host?

Answer:
We can use mac and IP parameter in the ***addHost()*** function as:

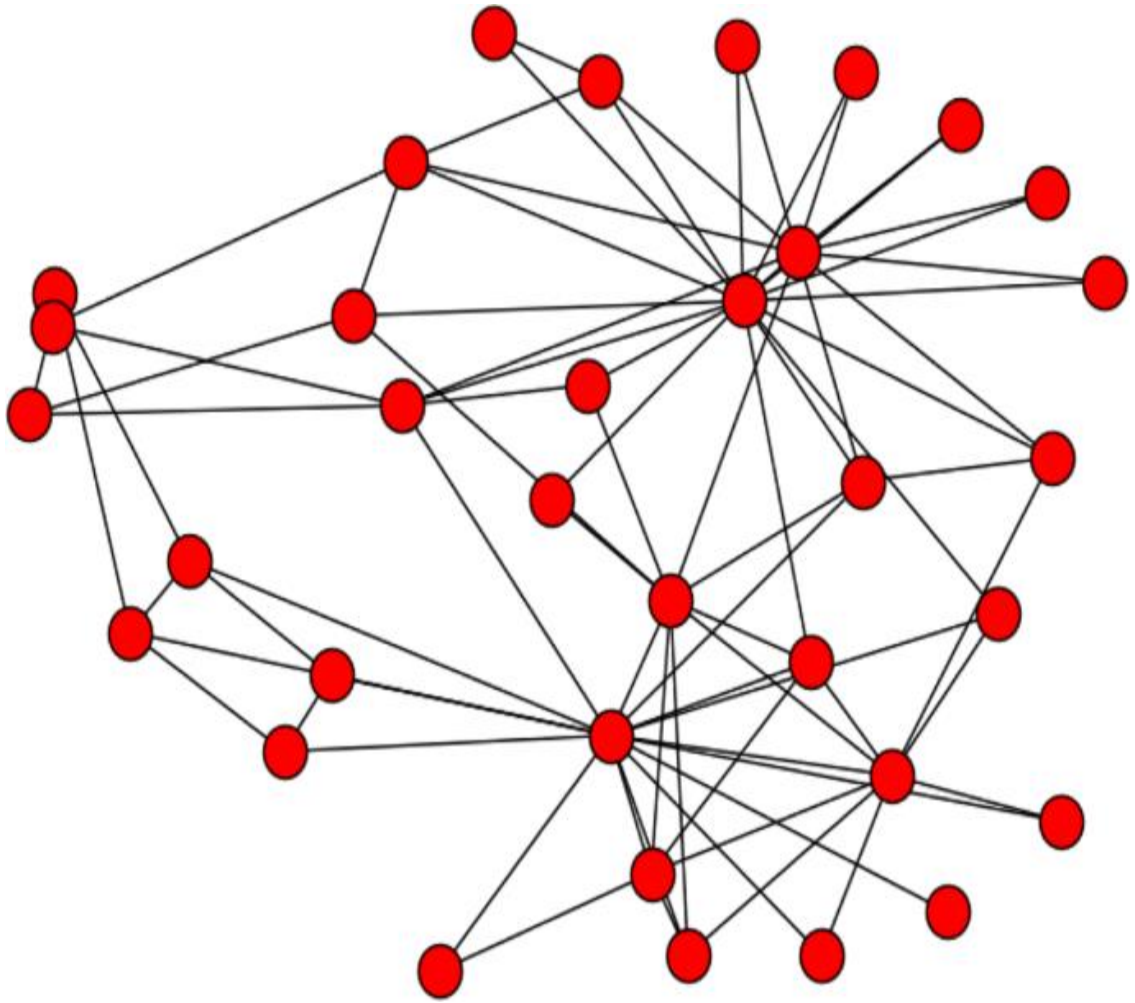
Host1 = self.addHost('h1', mac=’00:00:00:00:00:01’, ip=’192.168.1.100/24’)

Similarly, for other host also we can apply same method.

```
from mininet.topo import Topo
class MyTopo( Topo ):
    def __init__( self ):
        Topo.__init__( self )

        # Add hosts and switches
        Host1 = self.addHost( 'h1’, mac=’00:00:00:00:00:01’, ip=’192.168.1.100/24’)
        Host2 = self.addHost( 'h2' )
        Host3 = self.addHost( 'h3’ )

        Switch1 = self.addSwitch('s1')
        Switch2 = self.addSwitch('s2’)
        Switch3 = self.addSwitch(‘s3')

        # Add links
        self.addLink( Host1, Switch1 )
        self.addLink( Host2, Switch1 )
        self.addLink( Host3, Switch2 )
        self.addLink( Switch1, Switch2 )
        self.addLink( Switch2, Switch3 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

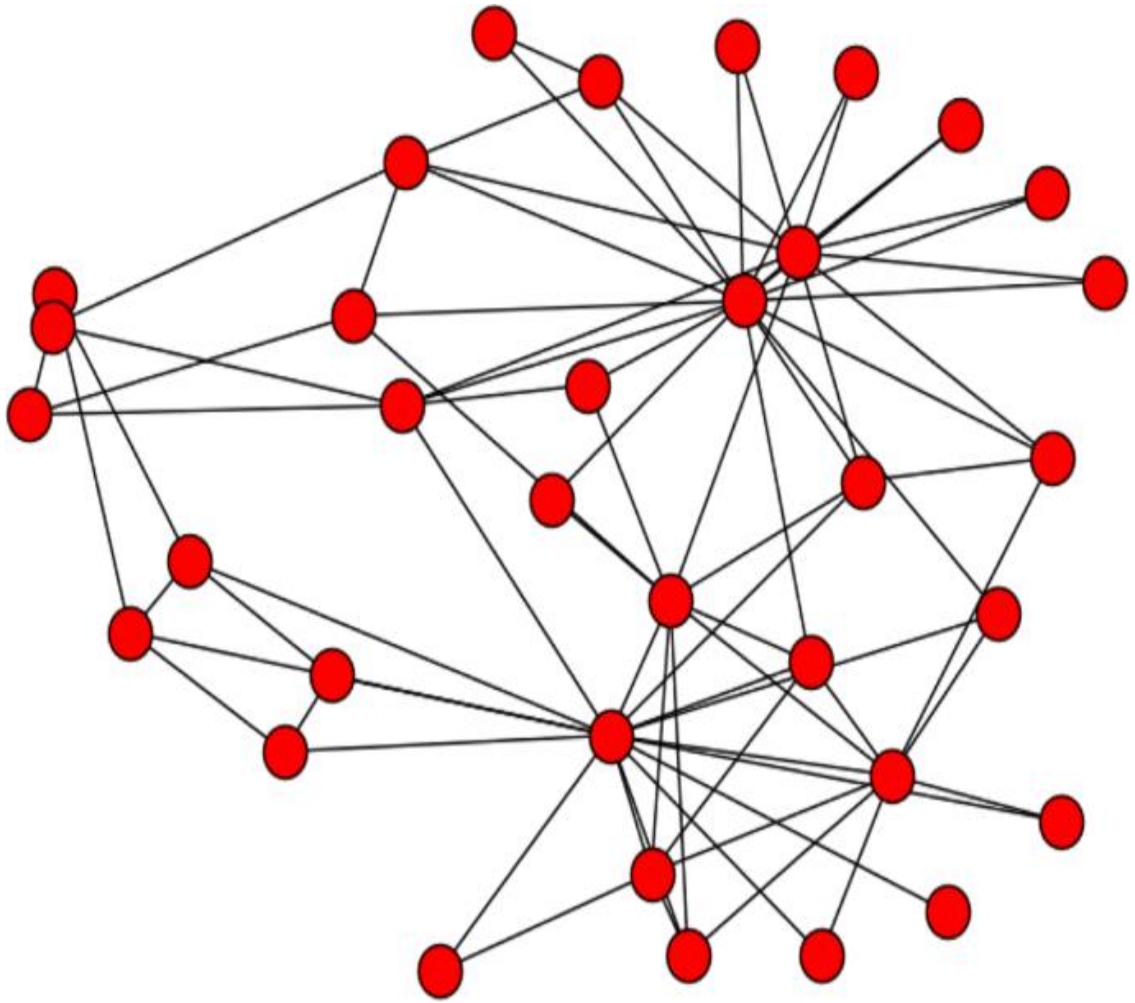
Complex Network Topologies

Q: How to build a more complex network topology where the number of host and switches are in thousands?



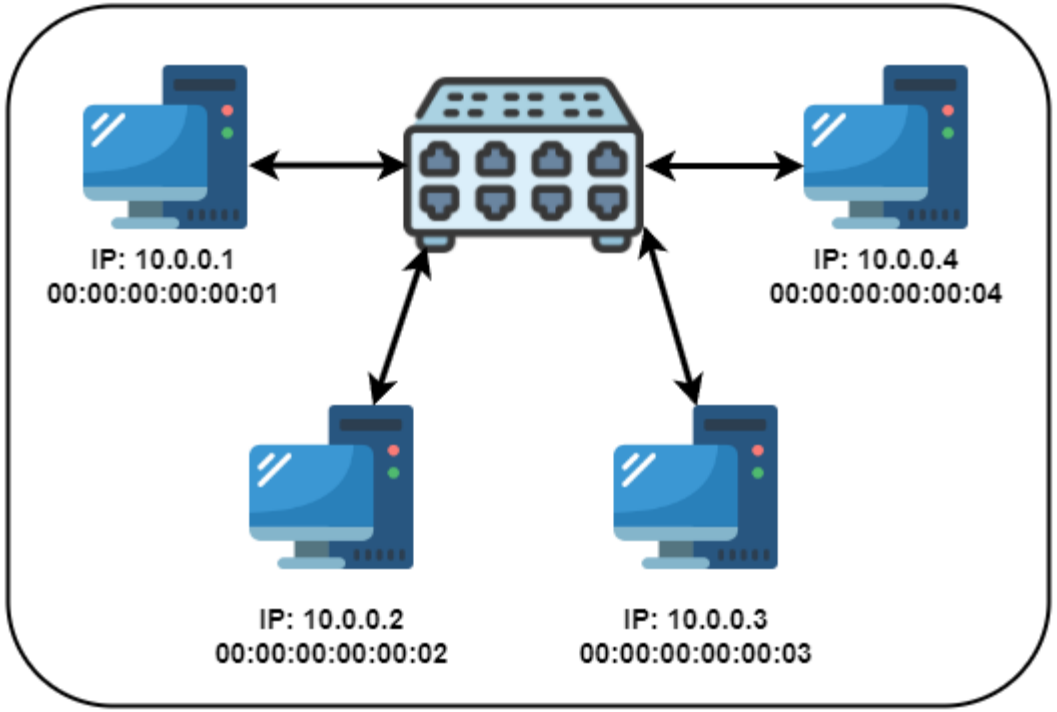
Complex Network Topologies

- **We can utilize the loop concept to build a more complex network.**



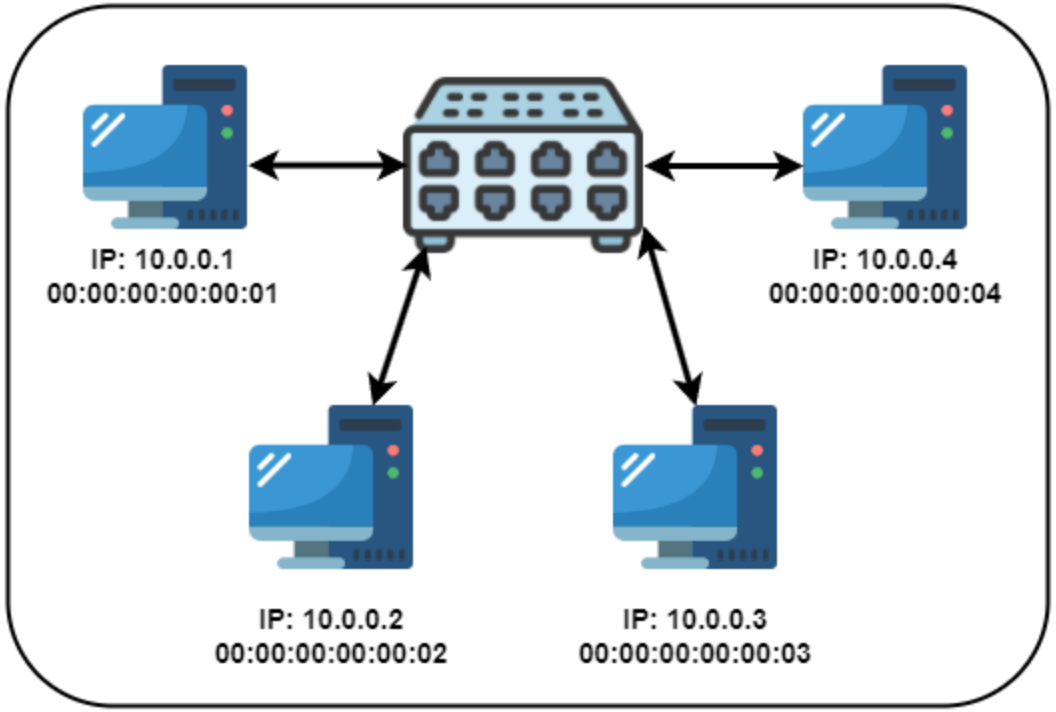
Complex Network Topologies

- **We can utilize the loop concept to build a more complex network.**
- **For the sake of example, we take an example of four host and one switch as shown in the figure.**



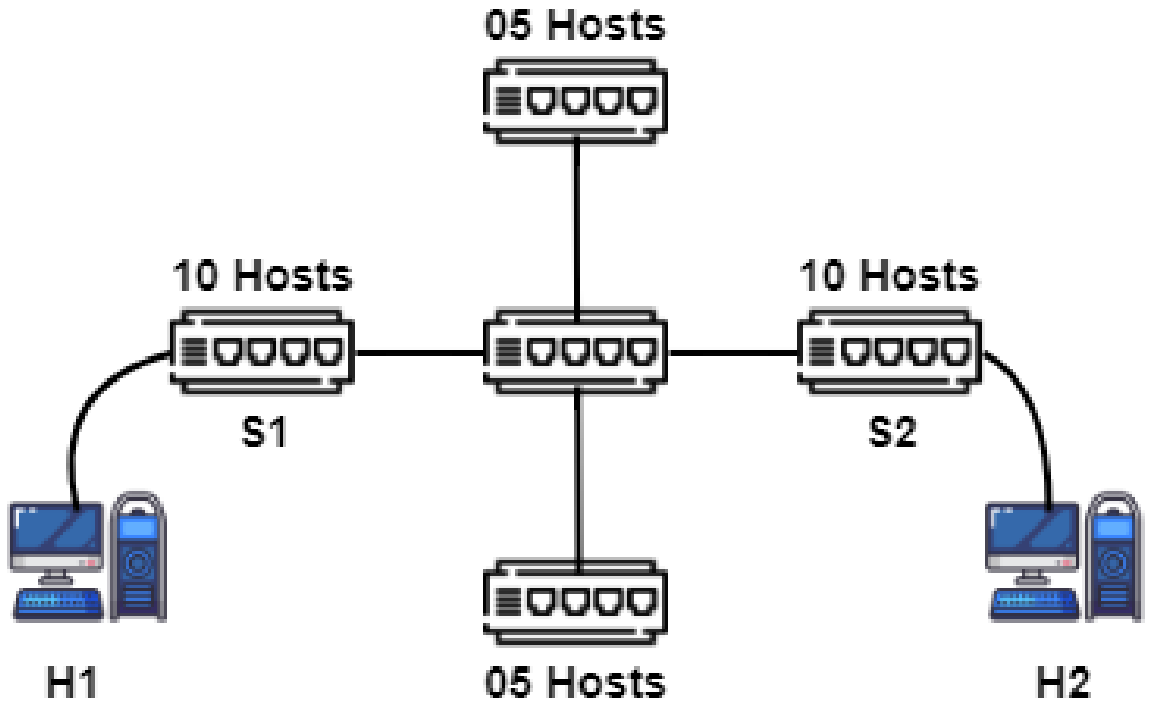
Complex Network Topologies

```
from mininet.topo import Topo

class MyTopo(Topo):

    def __init__(self, enable_all = True):
        Topo.__init__(self)

        no_of_host = 4
        no_of_switch = 1
        hosts = []
        switches = []

        for h in range(no_of_host+1):
            hosts.append(self.addHost('h%s' % (h+1)))

        for s in range(no_of_switch+1):
            switches.append(self.addSwitch('s%s' % (s+1)))

        self.addLink(hosts[0], switches[0])
        self.addLink(hosts[1], switches[0])

topos = {'mytopo': (lambda: MyTopo())}
```



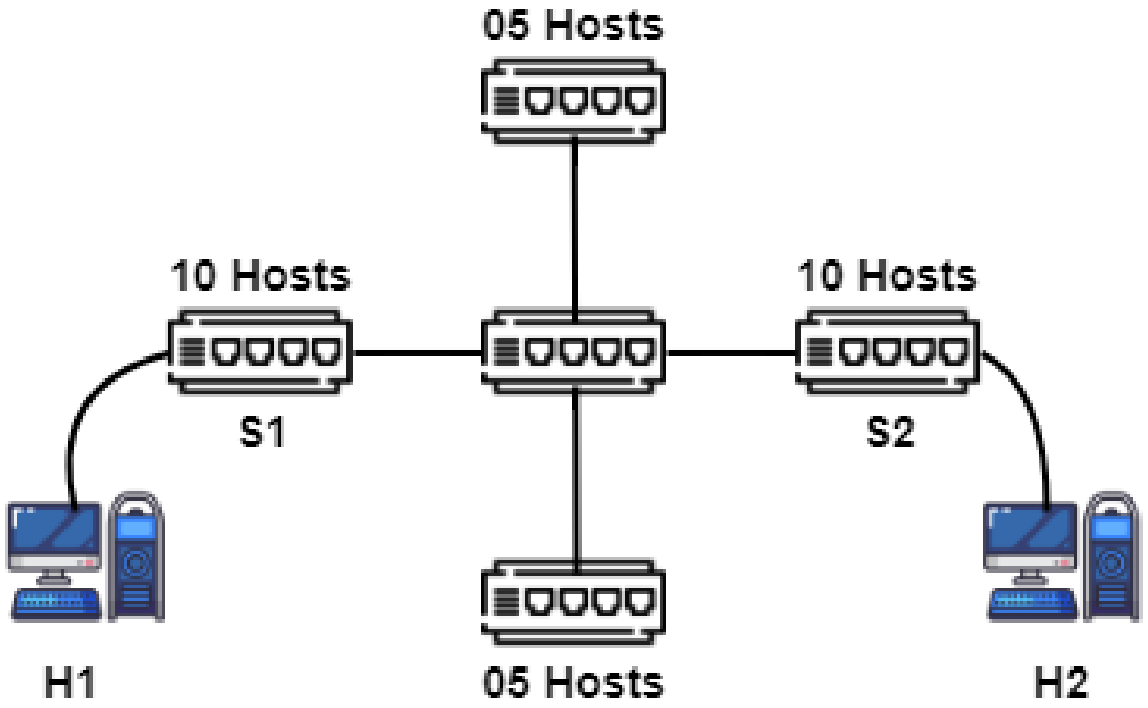
Complex Network Topologies

Question 1: Write a custom Mininet topology of the below diagram. Use the default IP and MAC addresses for the hosts. Create an ICMP stream between the host from H1 and H2.



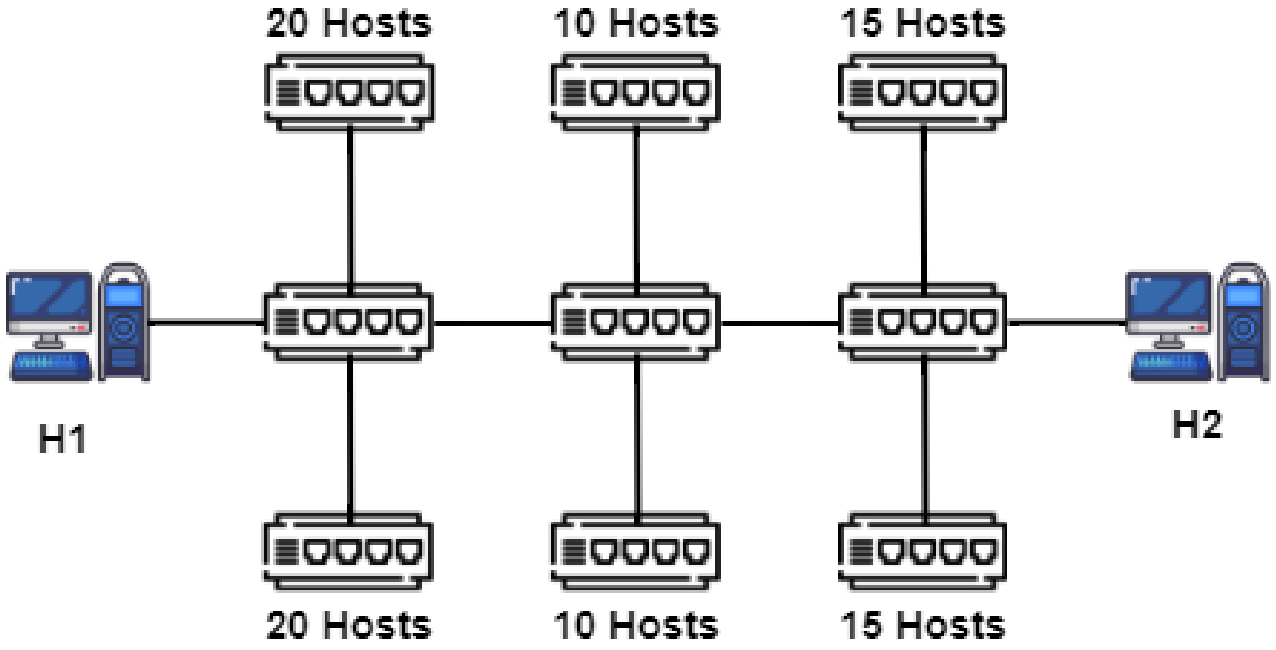
Complex Network Topologies

Question 2: Change the IP address of H1 to 192.168.127.100/24 and try to communicate with H2. Show the output.



Complex Network Topologies

Question 3: Write a custom mininet topology of the below diagram using for loop for host and switch creation. Use the IP address of Class C (192.168.127.0/24) for the hosts. Upon completion run the script and show the successful communication. Confirm the functionality of the IP address.



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

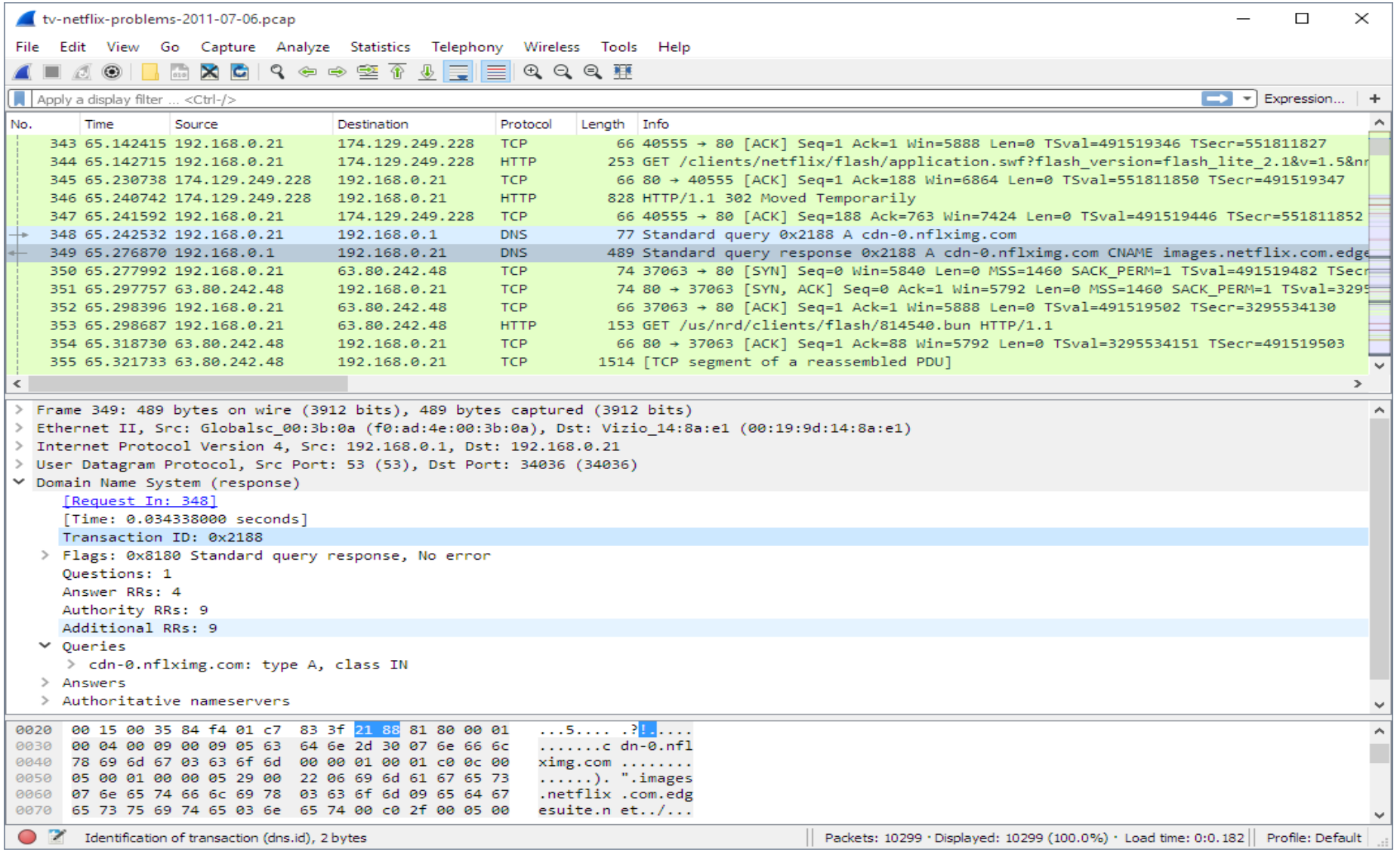
Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

Wireshark

Here are some things Wireshark does not provide:

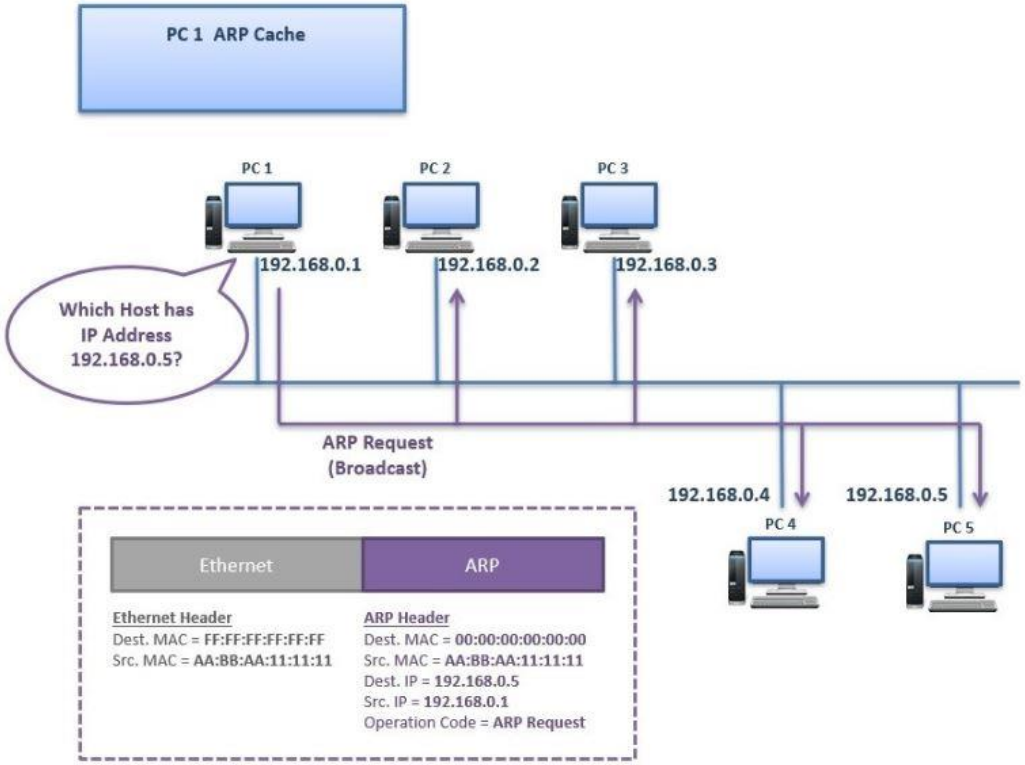
- Wireshark isn’t an intrusion detection system.
- Wireshark will not manipulate things on the network, it will only “measure” things from it.

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.



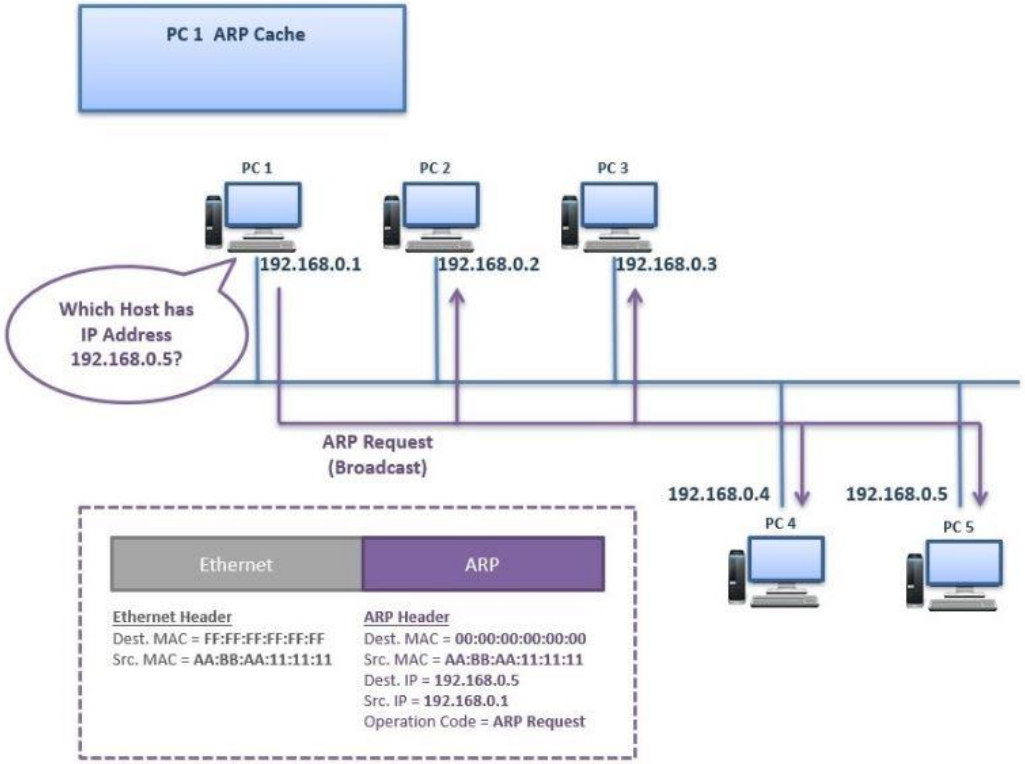
Address Resolution Protocol (ARP)

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

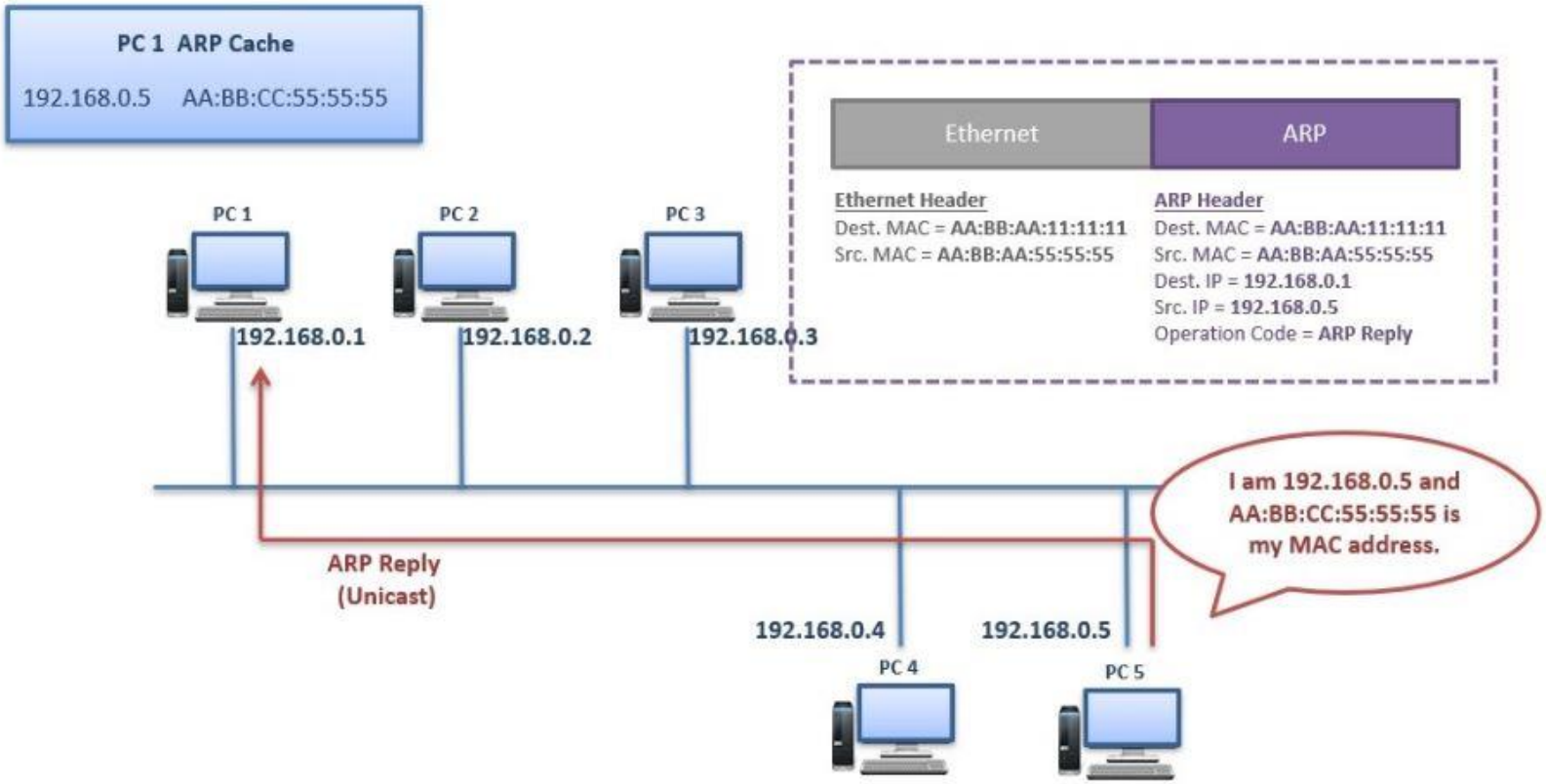
“Which Host has IP Address 192.168.0.5?”

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 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.



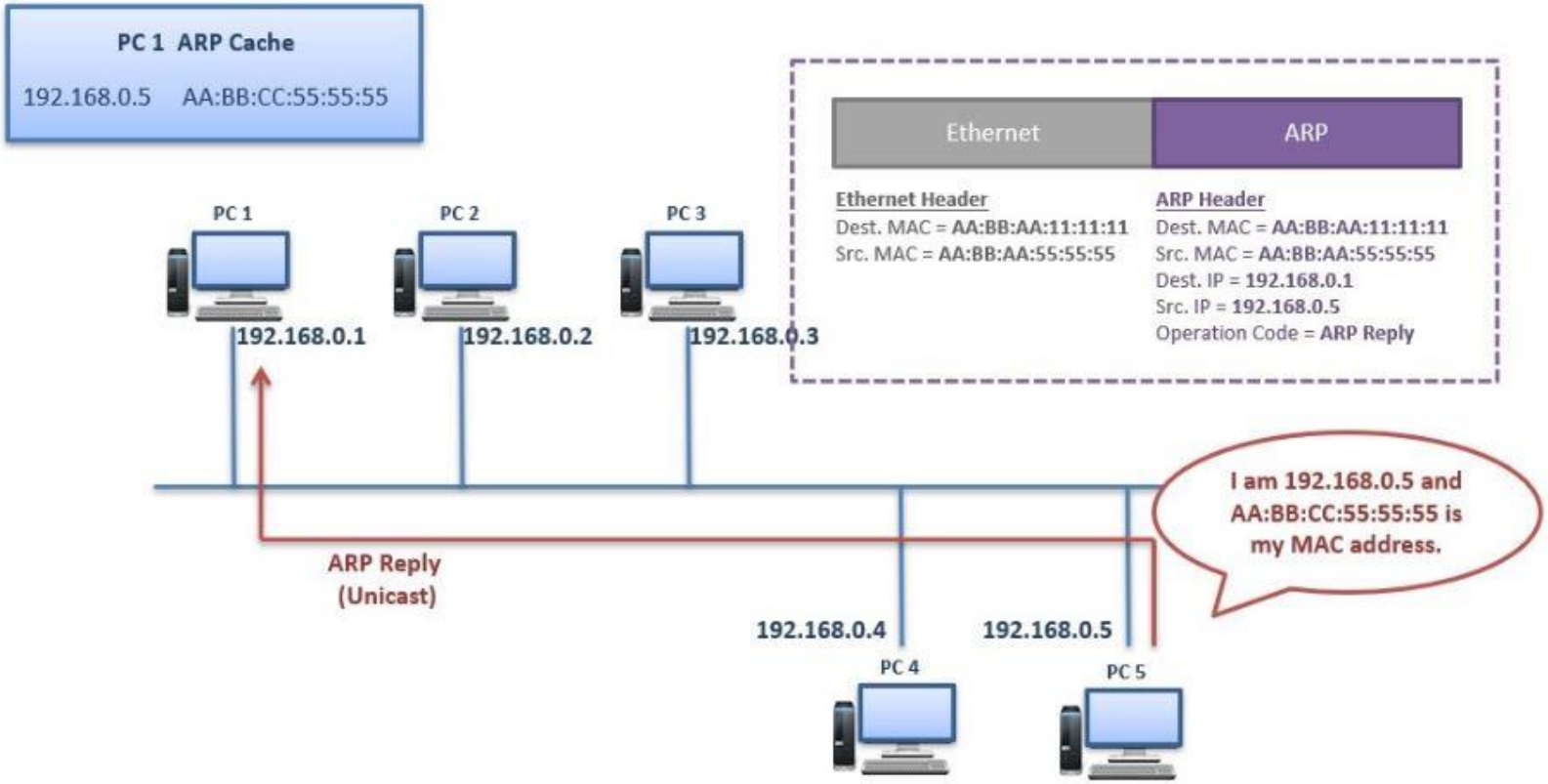
Address Resolution Protocol (ARP)



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

Transmission Control Protocol (TCP)

- TCP 3-way handshake or three-way handshake
- It is a process which is used in a TCP/IP network to make a connection between server and client.
- It is a three-step process that requires both the client and server to exchange synchronization and acknowledgement packets before the real data communication process starts.

- Syn use to initiate and establish a connection
- ACK helps to confirm to the other side that it has received the SYN.
- SYN-ACK is an SYN message from the local device and ACK of the earlier packet.
- FIN is used for terminating a connection.