Mininet 3+4 and Wireshark

Running GUI Apps through Mininet VM

Can you SSH into your VM?

- **Yes, I can SSH into my Mininet VM**
  - **Install and run X11 Server**
  - **Mac users:**
    - **XQuartz:** https://www.xquartz.org/
  - **Windows users:**
    - **Use WSL, Putty, or other X11 compatible SSH client (not ssh from Windows 10 cmd.ex)**
    - **VcXsrv:** https://sourceforge.net/projects/vcxsrv/
  - **Issues?**
    - **See here:** https://github.com/mininet/mininet/wiki/FAQ#x11-forwarding
- **No, I am using the VirtualBox console or the above didn't work**
  - **https://github.com/mininet/mininet/wiki/FAQ#can-i-run-a-guix11-application-within-a-mininet-host**
  - **Follow those instructions to run Wireshark inside the VirtualBox window.**

Software Defined Networking

SDN splits the Control and Data planes to allow programmatic control of networks

- Data Plane
  - Responsible for moving data from one part to another - the ‘flows’ in a switch
  - Needs to be very fast and low latency
- Control Plane
  - Responsible for deciding where data goes
  - It controls the Data plane

Why do we want or care about SDN?

Hopefully this picture sums it up

OpenFlow

SDN splits the Control and Data planes to allow programmatic control of networks

OpenFlow is an standard SDN protocol.

The protocol can be and is implemented in real hardware.

- Used in datacenter switches
- Alternative to using Vendor-specific configuration tools such as Cisco IOS CLI

It can also be implemented in software-based virtual networks.

- Mininet

What is Mininet?

Software that creates a virtual network using process-based abstraction

Each process runs in its own virtual network namespace

- Has its own virtual network hardware as well as IP’s and MAC addresses

These virtual networks are used extensively in the cloud as well as for container networking (Docker!)

SDN in Mininet

Data Plane:

- OpenVSwitch (https://www.openvswitch.org/)
  - This is an OpenFlow compatible Virtual Switch
  - It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols

Control Plane

- OpenFlow Controller which you implement in Mininet 2, 3, and 4.
- In our case we are using the Python-based Pox OpenFlow Library.
  - There are plenty of OpenFlow libraries in other languages

Mininet 1

You created a virtual network topology containing multiple hosts and a switch using Mininet.

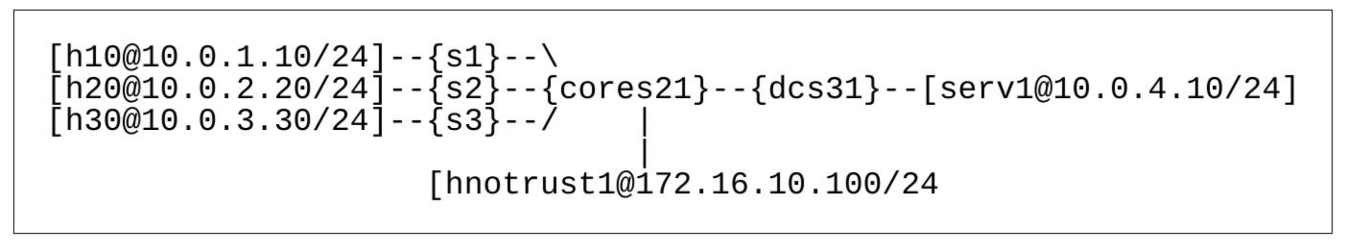
Mininet 2

You implemented a basic Firewall using Pox-based Openflow controller

Your controller, on startup, installed data-flow rules (ofp_flow_mod) into the Switch to:

- Flood ARP packets
- Flood ICMP packets
- Drop all other (IPv4) packets

Mininet 3



```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/    |
                               |
                    [hnotrust1@172.16.10.100/24
```

You implement a controller for a more complex virtual network topology
There are now multiple switches connected to a central router (cores21)
The switches - s1, s2, s3, and dcs31 - should be very simple and can just flood traffic
The router, cores21, cannot flood all ports and should use a specific port for the destination subnet.

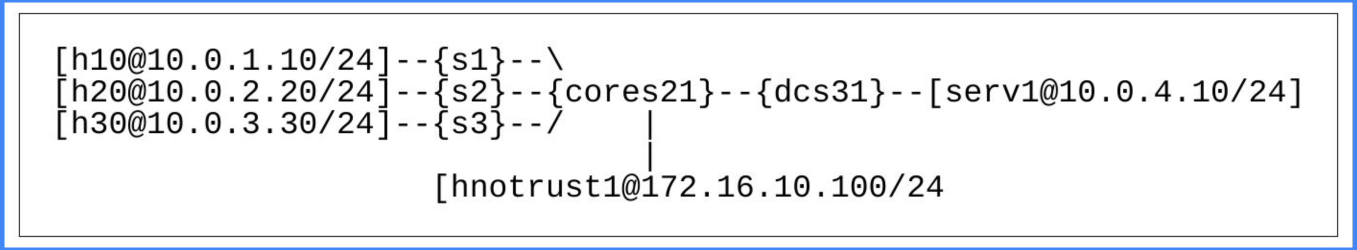
In mininet cli:

```
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
...
s1 lo:  s1-eth1:h10-eth0 s1-eth2:cores21-eth1
mininet> ports
cores21 lo:0 cores21-eth1:1 cores21-eth2:2 cores21-eth3:3 cores21-eth4:4 cores21-eth5:5
...
```

You will need to use the specific individual switch port numbers (not OFPP_FLOOD) for cores21

- of.ofp_action_output(port = PORTNUM)
- Can be done dynamically
  - Easier and acceptable to hardcode (not acceptable for part4)

Mininet 4



```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24
```

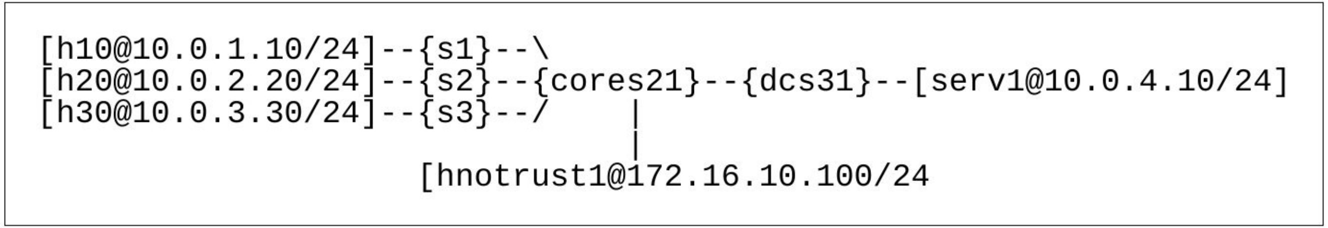
Similar topology to Mininet 3, but more intelligent, real routing.

Mininet 3 ‘brute-forces’ the network by forwarding packets between switches and routers.

In Mininet 4, you implement an actual router which the hosts talk to as a ‘gateway’:

- cores21 functions as an L2 switch in part3
  - Forwards the unmodified packets to the correct destination
- cores21 functions as an L3 router in part4
  - L3 routers will change MAC addresses when routing between subnets
  - Mininet hosts will be expecting this type of L3 functionality
  - Warning: hosts will ignore packets that don’t have the correct MAC addresses

Mininet 4

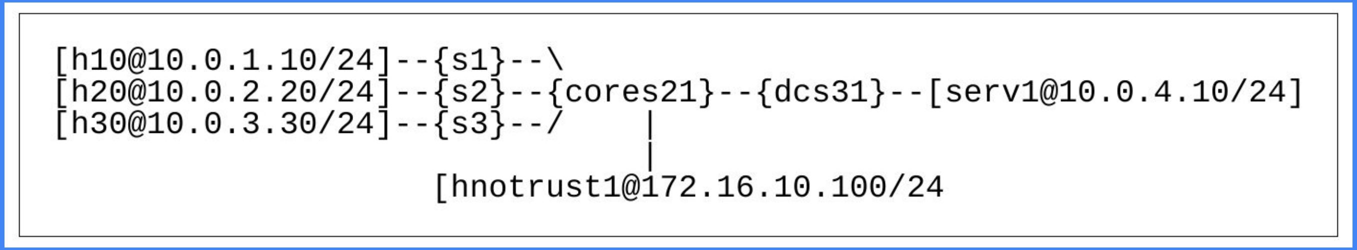


```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24
```

The Mininet 4 Gateway needs to do the following:

1. Proxy ARP messages for destinations that are outside the local subnet by responding with the gateway's MAC address (arbitrarily chosen by you in OpenFlow controller software).
   a. ARP Request messages should be sent to OpenFlow controller's Handle_PacketIn method
   b. The OpenFlow controller should then generate an ARP reply telling the host to use the gateway's address for packets destined to that subnet
   c. Now when the host tries to reach that subnet, it will direct its messages to the gateway's MAC address.
2. Learn host IP's from the received ARP messages/broadcasts
   a. Hosts broadcast 'whohas' ARP requests containing the host's own IP address and MAC address
   b. The gateway learns the IP's from requests and replies to them with its own MAC.
   c. It can install rules that match based on 'destination address' here
   d. Note that communication across subnets will not work until the switch has learned both hosts or subnets
   e. i.e. H10 <-> H20 will not work until both H10 and H20 both individually try to reach each other, teaching the gateway their IP address and port.

Mininet 4



```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24
```

The Mininet 4 Gateway needs to do the following:

3. Route IP packets across subnets

- Will need to change the Source and Destination MAC addresses
- For Incoming IP packets to the gateway
  - ○ Source MAC is the original host that sent the packets
  - ○ Destination MAC is the gateway
    - ■ Packet’s destination IP address is the actual destination host
    - ■ The packet’s destination MAC address is the gateway
- Outgoing IP packets from gateway
  - ○ Source MAC is the gateway
  - ○ Destination MAC is the MAC address associated with destination IP
    - ■ Hint: ofp_flow_mod actions: of.ofp_action_dl_addr

Wireshark in Mininet

Open Wireshark from Mininet SSH session with X11 or from Mininet GUI if you installed this

- sudo wireshark &
  - `sudo` needed to see all network interfaces
  - `&` will run the process in the background, preventing it from stealing your terminal
- Capture on ‘**Loopback: lo**’ for packets from the controller
  - Can also capture packets sent by mininet devices: ‘**s1**’, ‘**s1-eth1**’,....
    - These will not appear in Wireshark unless you run the mininet topology **before** Wireshark
- Capture on ‘**all**’ to see all packets - will need to filter to avoid packets from SSH
  - In part2, mininet hosts are on subnets 10.0.1.0/24 and 10.0.0.0/24
  - Wireshark can filter for openflow packets with ‘of’
  - of || arp || ip.addr == 10.0.1.0/24 || ip.addr == 10.0.0.0/24
    - This will filter for openflow packets, arp, and packets from the mininet hosts
  - Virtualbox NAT uses subnet 10.0.2.0/24, which is the same as h2 in parts 3+4
    - Filter for the specific IP address of h2 (10.0.2.20) instead in parts 3+4