Section 6: Mininet II

CSE 461 Computer Networks
Happy Lunar New Year!

Technically tomorrow in PST, but it’s already new year in Asia!
Hopefully part2 wasn’t too bad...
Part 3

- You can hardcode who-is-where in cores21_setup.
- Run links in the Mininet console to see who’s where.
- If your pingall fails, make sure that you flood ARP. (Why?)

```
[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]
```
Using Wireshark with the Mininet VM (Demo at the end)

- **In host (your physical computer/CSE VDI machine):**
  - Install X Window Server: XQuartz (macOS host) / Xming or VcXsrv (Windows host)
  - For macOS, you *might* need this near the top of your ~/.ssh/config (try if it doesn’t work w/o it):
    ```
    Host *
    XAuthLocation /usr/X11/bin/xauth
    ```

- **In VM (Vagrant/VMware/EC2 instance):** [ssh -X into your VM if not using vagrant]
  - Install Wireshark: `sudo apt install wireshark`
  - Launch your controller (another terminal): `sudo ~/pox/pox.py misc.part3controller`
  - Magic command req’d for Vagrant: `sudo xauth add $(xauth list $DISPLAY)`
  - Launch Wireshark as root: `sudo wireshark &`
  - You should be able to see the ethX interfaces for your switches
Alternatively...

- If Wireshark doesn’t work for you, you can also simply dump packet content in _handle_PacketIn by printing out `packet.dump()` (I did that when I took the class and it was good enough, although Wireshark somehow looks cooler.)
Part 4

- Must not hardcode who-is-where.
- Learn by backward learning --- learn who-is-where when we hear from them.
I want to ping 10.0.2.20. That’s not in my subnet, so I know that should go through 10.0.1.1.

\[
\begin{align*}
[h10@10.0.1.10/24] & \rightarrow \{s1\} \\
[h20@10.0.2.20/24] & \rightarrow \{s2\} \rightarrow \{cores21\} \rightarrow \{dcs31\} \rightarrow \{serv1@10.0.4.10/24\} \\
[h30@10.0.3.30/24] & \rightarrow \{s3\} \\
& \rightarrow \{hnotrust1@172.16.10.100/24\}
\end{align*}
\]
ARP REQUEST:
Who is 10.0.1.1?
Tell 10.0.1.10.

[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]
I just got an ARP request from 10.0.1.0/24 through port 1. So in the future, I will remember to forward traffic to 10.0.1.0/24 through port 1. [Installs a ofp_flow_mod rule]
Part 4 - h10 ping h20

I'm going to handle traffic for s1. 😐

```
[h10@10.0.1.10/24]--{s1}--\n[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]
```
Part 4 - h10 ping h20

[\texttt{h10@10.0.1.10/24}]--\{s1}\--\textbackslash
[\texttt{h20@10.0.2.20/24}]--\{s2}\--\{\texttt{cores21}\}--\{\texttt{dcs31}\}--[\texttt{serv1@10.0.4.10/24}]
[\texttt{h30@10.0.3.30/24}]--\{s3}\--/
\hspace{1cm}|
\hspace{1cm}|
\hspace{1cm}[\texttt{hnotrust1@172.16.10.100/24}]

10.0.1.1 is at de:ad:be:ef:ca:fe (I just made that up, but I replied so that's me 🙋).
Ok, I got the ARP reply. I think cores21 has 10.0.1.1. In the future, I will send out-of-network traffic through cores21.
Part 4 - h10 ping h20

Ping 10.0.2.20

[h10@10.0.1.10/24]--{s1}--\n[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]
Part 4 - h10 ping h20

I just got ICMP traffic to 10.0.2.20, but I don’t know where it’s at. I’ll just drop it.

```
[h10@10.0.1.10/24]--{s1}--\n[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]
```
Part 4 - h10 ping h30

[**okay.**]

[**times out**]

```
[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]
```
I want to ping 10.0.1.10. That’s not in my subnet, so I know that should go through 10.0.2.1.
Part 4 - h20 ping h10

ARP REQUEST:
Who is 10.0.2.1?
Tell 10.0.2.20
I just got an ARP request from 10.0.2.20/24 through port 2. So in the future, I will remember to forward traffic to 10.0.2.20/24 through port 2. [Installs a `ofp_flow_mod` rule]
ARP REPLY:
10.0.2.1 is me,
de:ad:be:ef:ca:fe.
Ok, I got the ARP reply. I think cores21 has 10.0.2.1. In the future, I will send out-of-network traffic through cores21.
Part 4 - h20 ping h10
I just got ICMP traffic to 10.0.1.10. My rules tell me to forward it thru port 1.
I got the ICMP request. I'll respond.
I just got ICMP traffic to 10.0.2.20. My rules tell me to forward it thru port 2.
Part 4 - h20 ping h10

😊

noice
Part 4 Summary

- *cores21* will respond to all ARP requests, claiming to be every sX, so it can forward all the IP/ICMP traffic.
- Once *cores21* knows where each host is, it will install a rule to forward IP traffic to that host through that port. (But *don’t install duplicate rules*, b/c we don’t want the rule table to grow with pings.)
- Therefore, pings to a host will always fail until *cores21* hears from that host.
- What will the output of *pingall* look like? What if we run *pingall* again?
Q&A, Extra OH