Section 6: Mininet II

CSE 461 Computer Networks

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 or Cygwin X(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
```

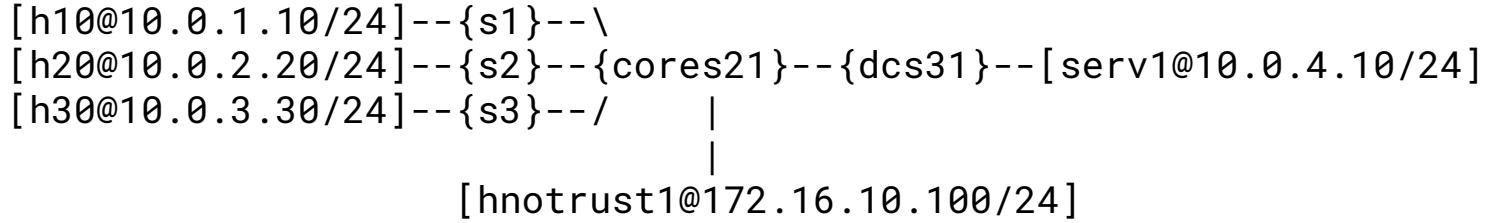
- In VM (Vagrant/VMware/EC2 instance): [`ssh -Y` into your VM if not using vagrant]
  - Install Wireshark: `sudo apt install wireshark`
  - Launch your controller (another terminal): `sudo ~/pox/pox.py misc.part3controller`
  - Magic command req'd for Vagrant: `sudo xauth add $(xauth list $DISPLAY)`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

Alternatively...

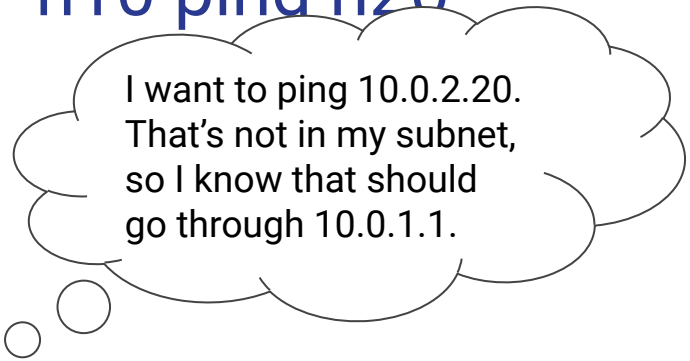
- If Wireshark doesn’t work for you, you can also simply dump packet content in `_handle_PacketIn` by printing out `packet.dump()` (I did that when I took the class and it was good enough, although Wireshark somehow looks cooler.)

Part 4

- Must not hardcode who-is-where.
- Learn by backward learning --- learn who-is-where when we hear from them.

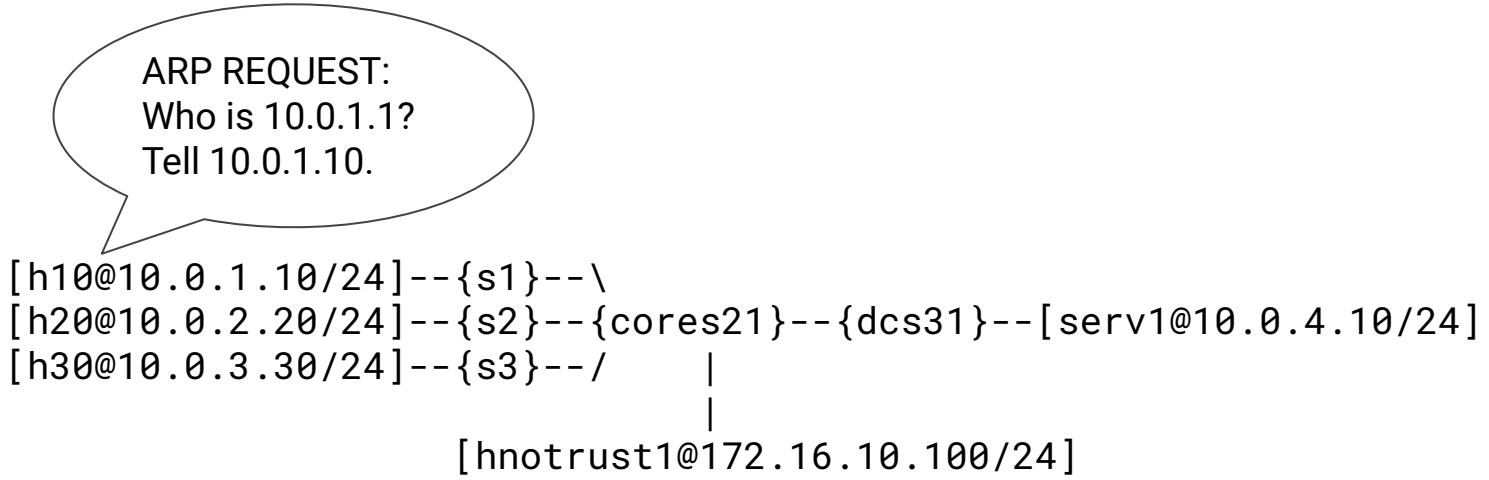


Part 4 - h10 ping h20

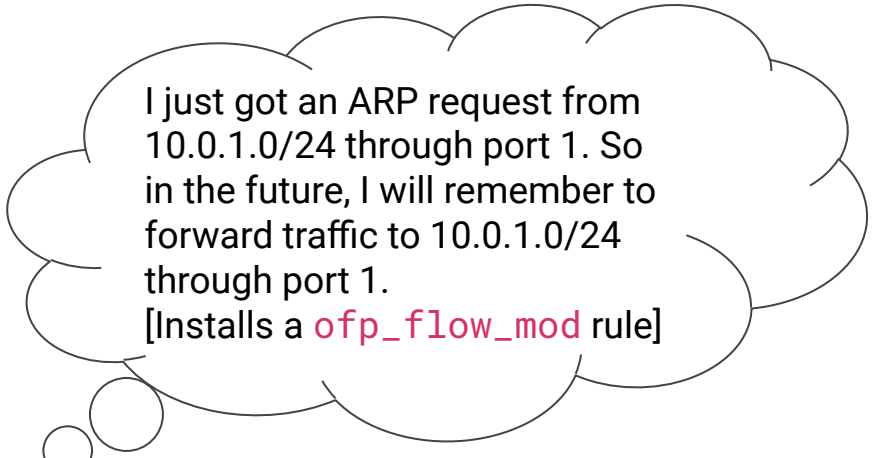



```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                  [hnotrust1@172.16.10.100/24]
```

Part 4 - h10 ping h20

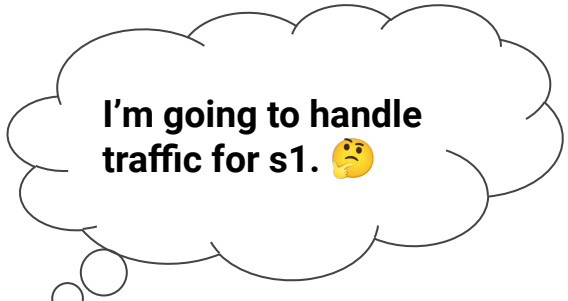


Part 4 - h10 ping h20



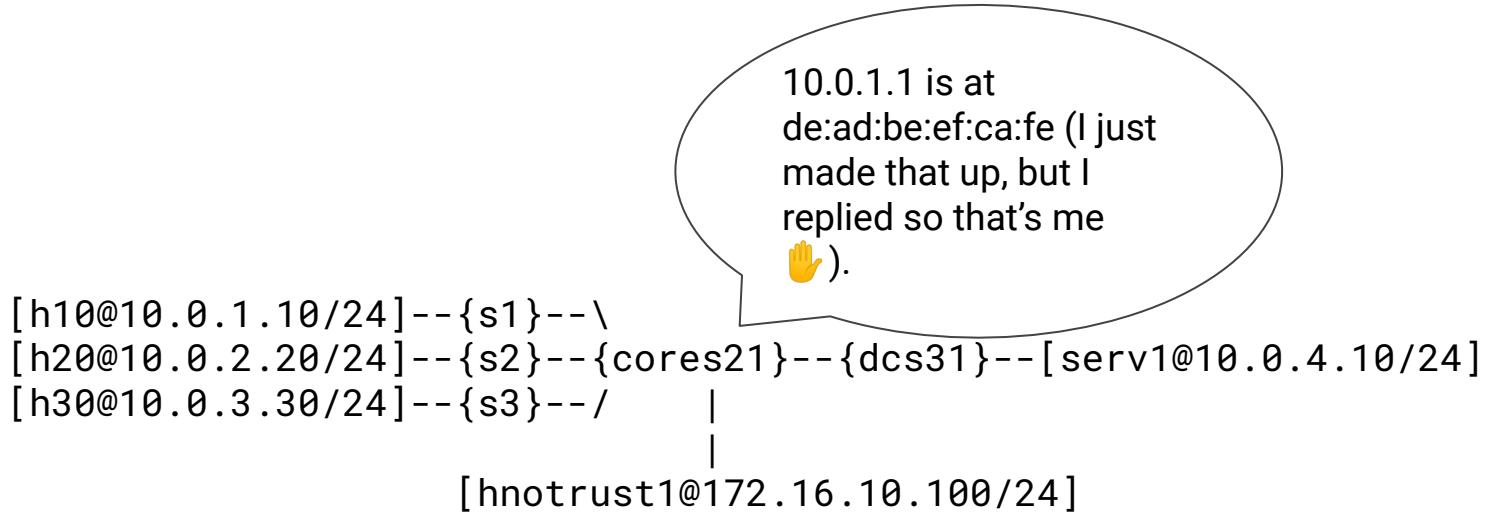

```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                  [hnotrust1@172.16.10.100/24]
```

Part 4 - h10 ping h20



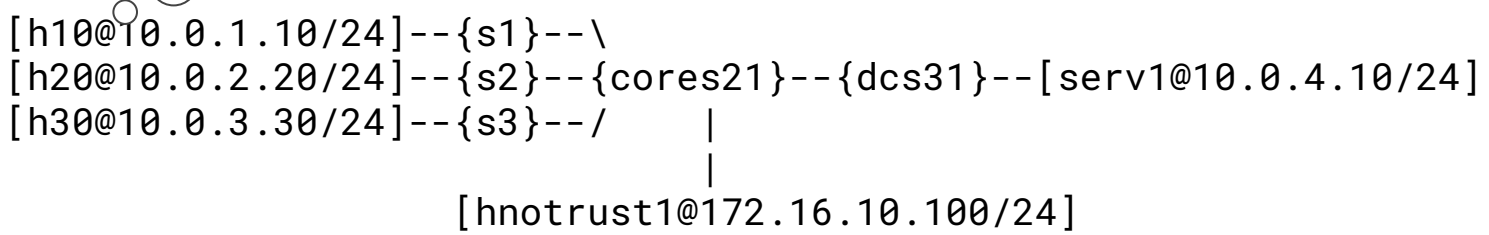

```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24]
```

Part 4 - h10 ping h20

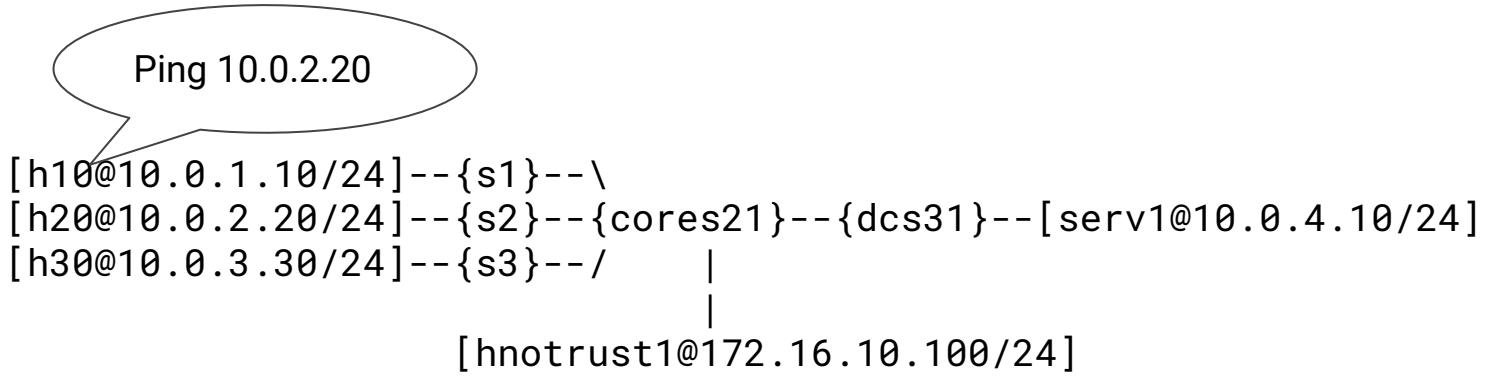


Part 4 - h1

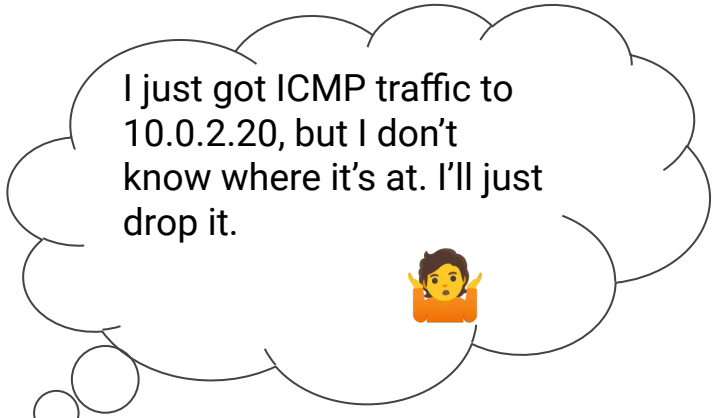
Ok, I got the ARP reply. I think cores21 has 10.0.1.1. In the future, I will send out-of-network traffic through cores21.



Part 4 - h10 ping h20

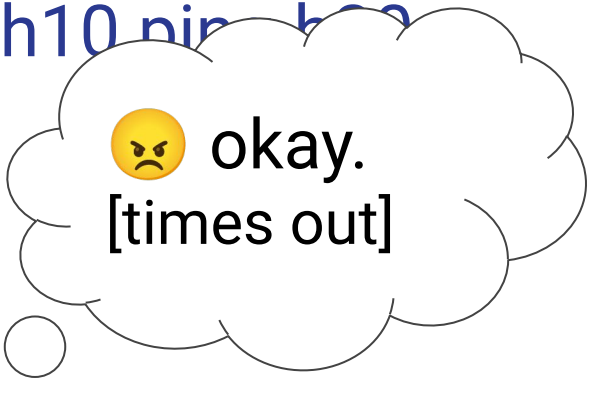


Part 4 - h10 ping h20



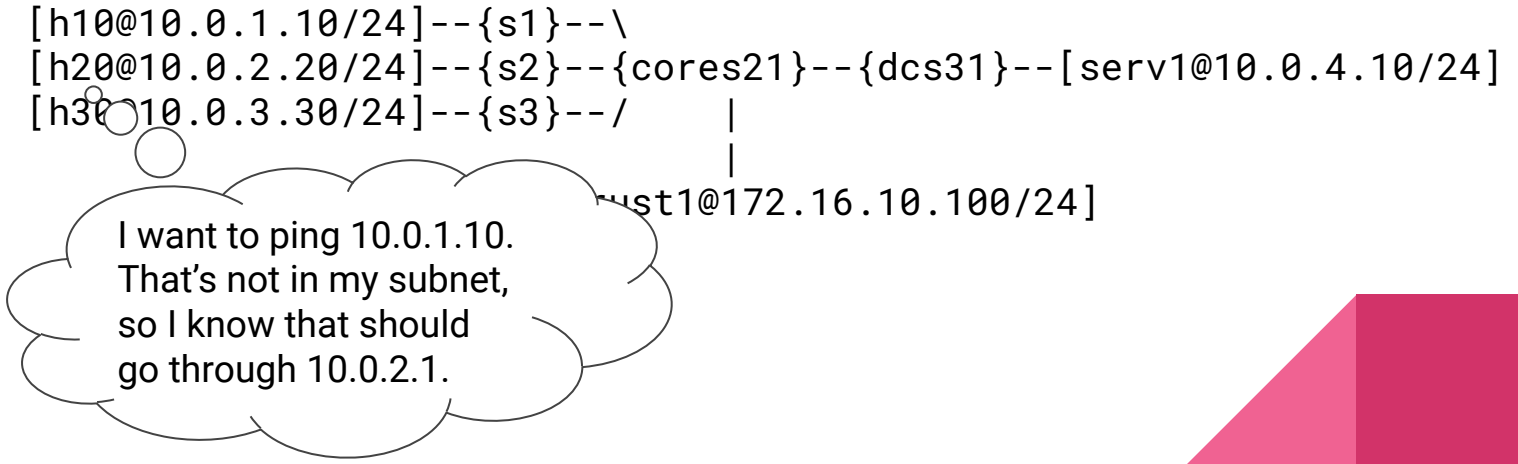

```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24]
```

Part 4 - h10 pin[illegible]

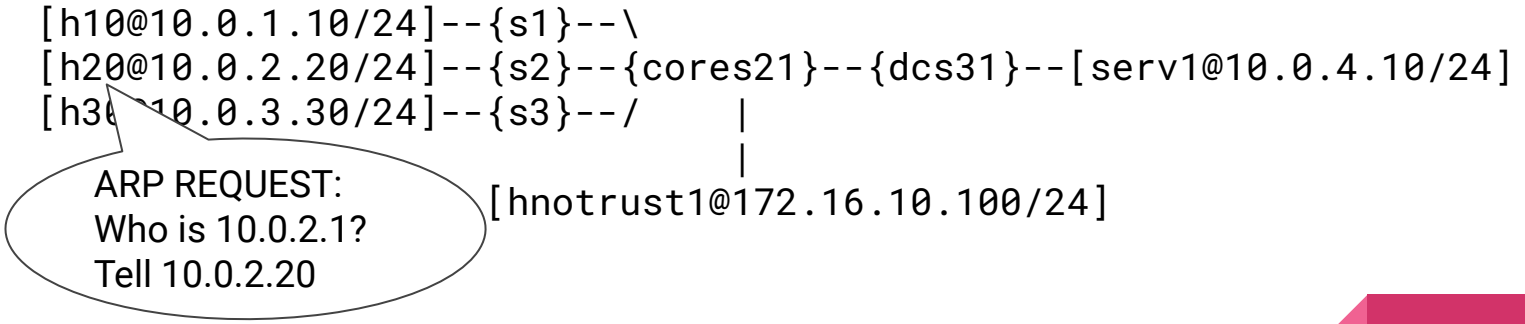



```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24]
```

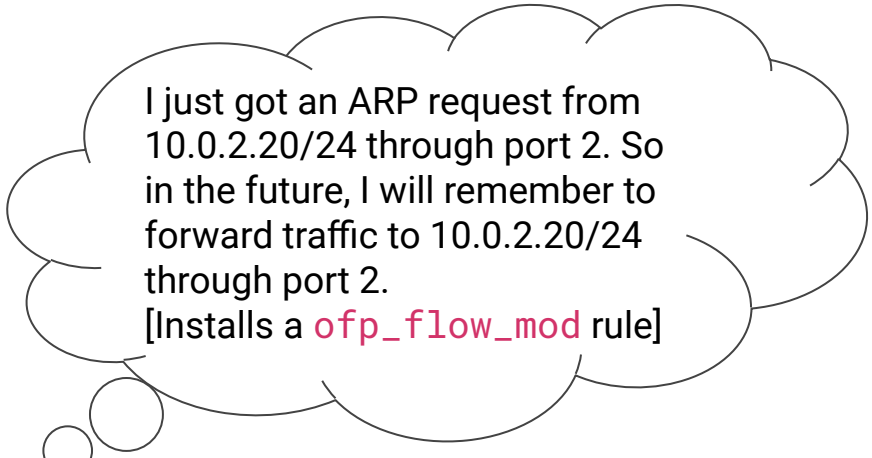
Part 4 - h20 ping h10



Part 4 - h20 ping h10

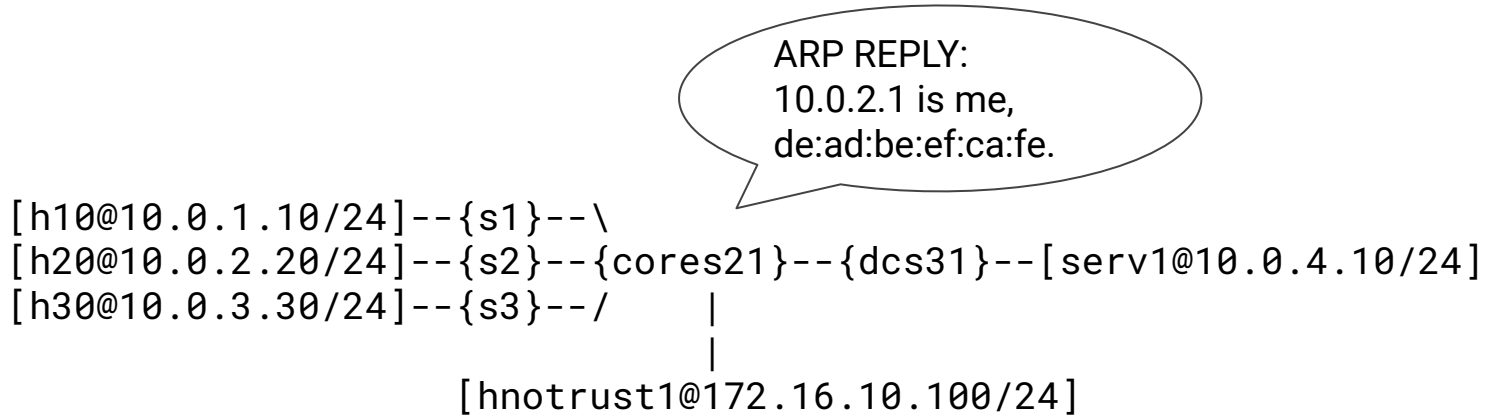


Part 4 - h20 ping h10

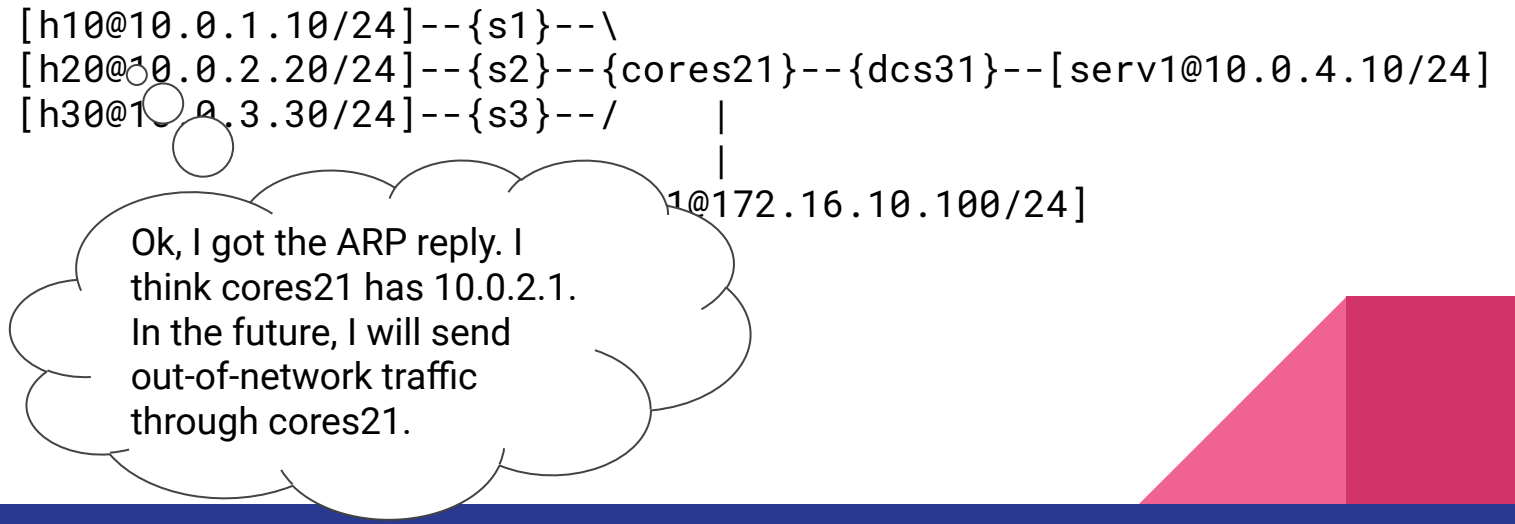



```
[h10@10.0.1.10/24]--{s1}--\
[h20@10.0.2.20/24]--{s2}--{cores21}--{dcs31}--[serv1@10.0.4.10/24]
[h30@10.0.3.30/24]--{s3}--/     |
                                |
                    [hnotrust1@172.16.10.100/24]
```

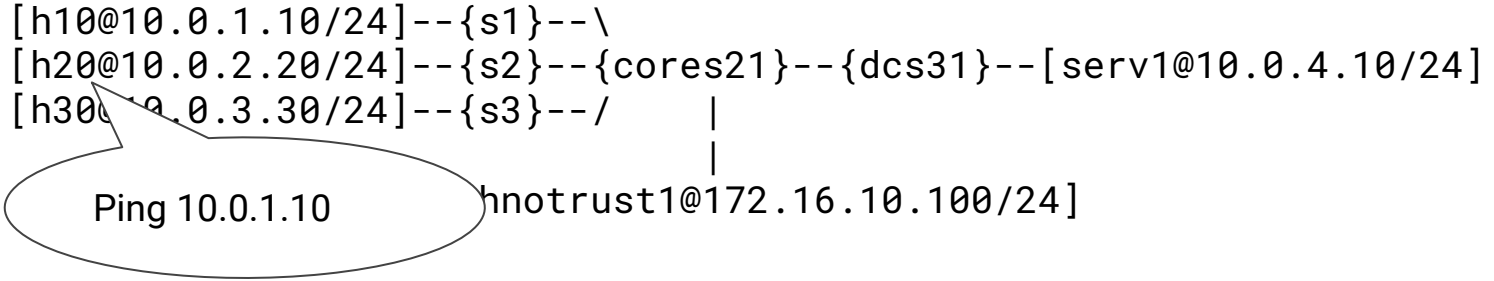
Part 4 - h20 ping h10



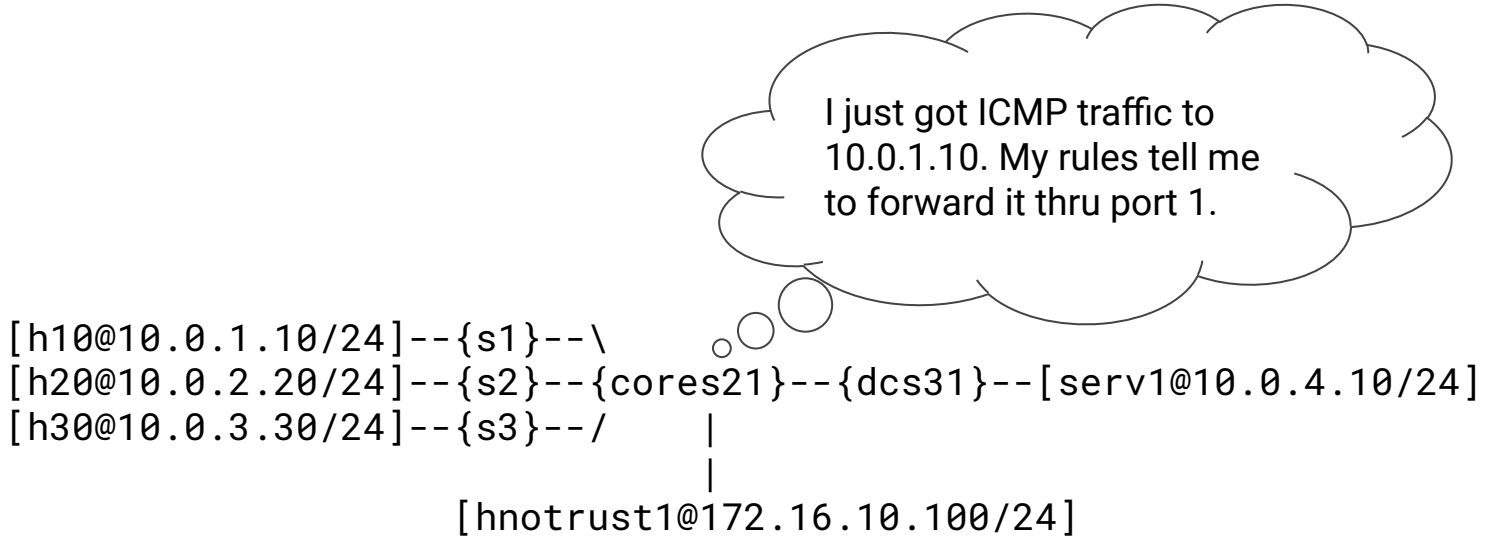
Part 4 - h20 ping h10



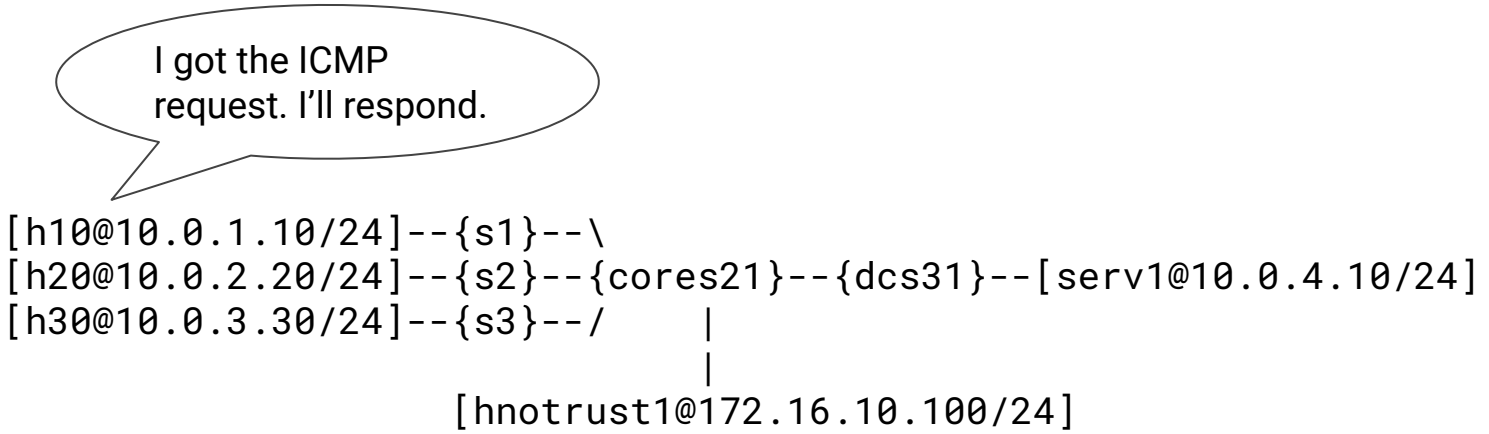
Part 4 - h20 ping h10



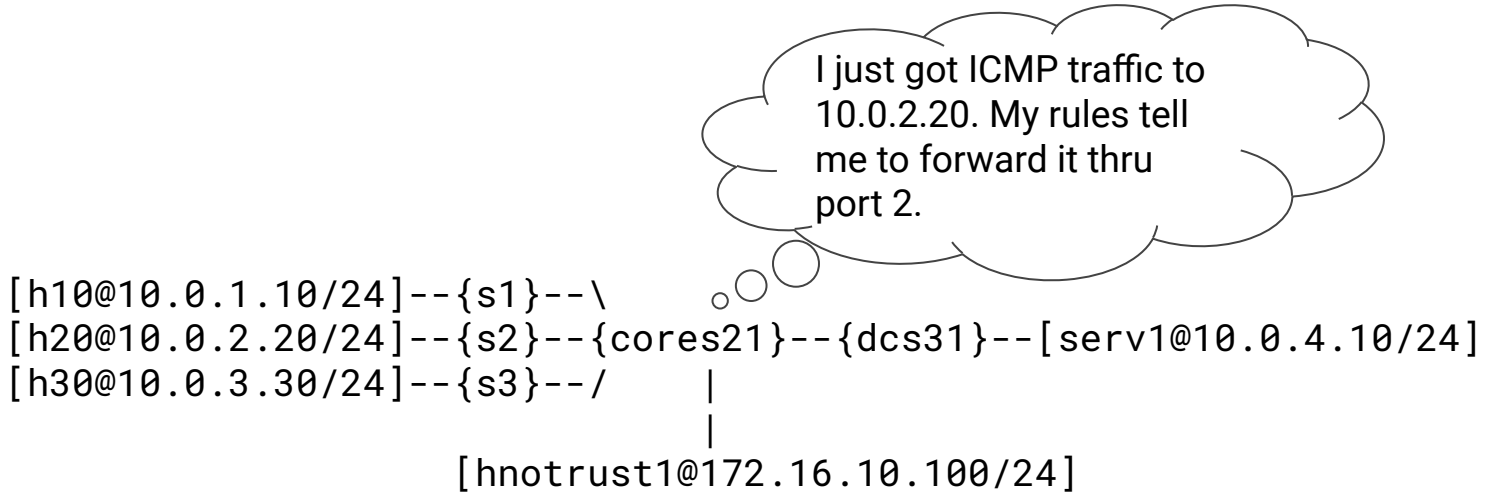
Part 4 - h20 ping h10



Part 4 - h20 ping h10

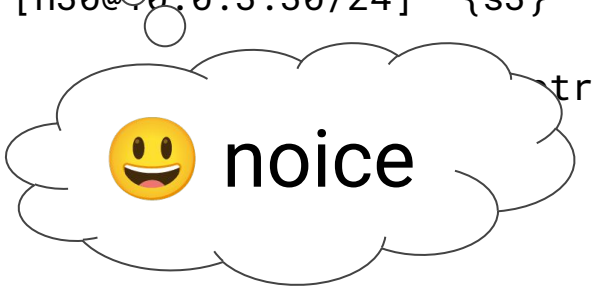


Part 4 - h20 ping h10



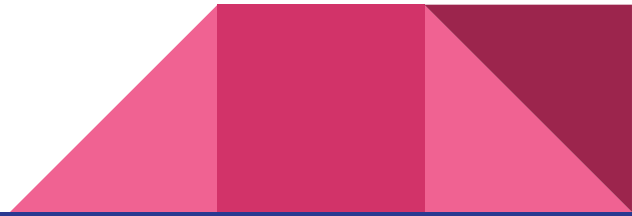
Part 4 - h20 ping h10

```
[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}--/     |
                                |
                          trust1@172.16.10.100/24]
```



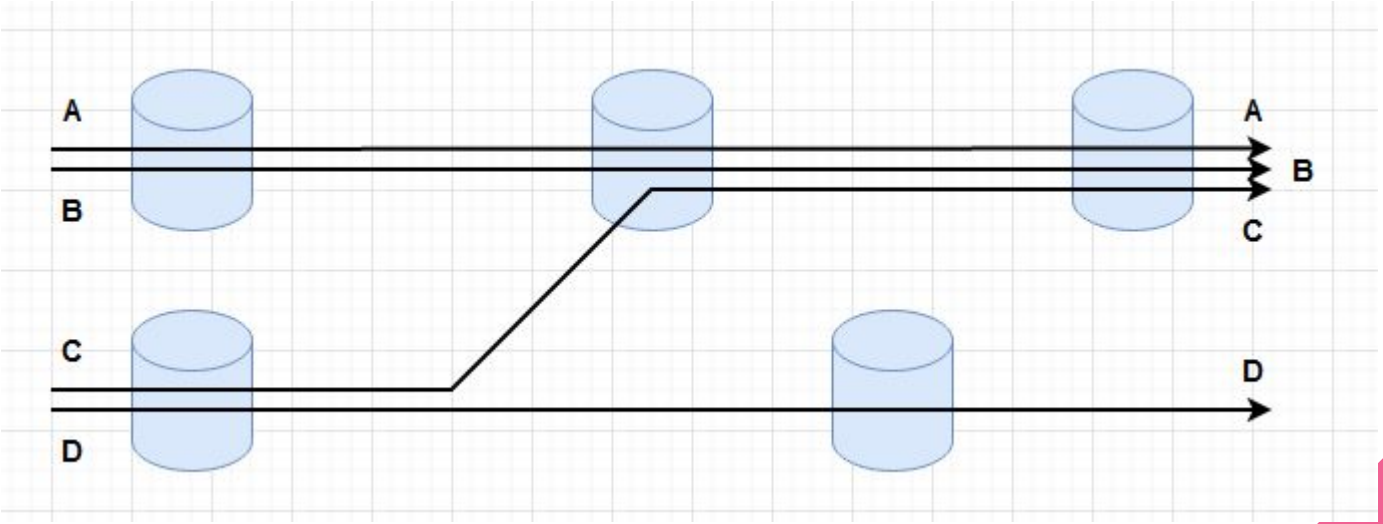
Part 4 Summary

- `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?



Extra exercise

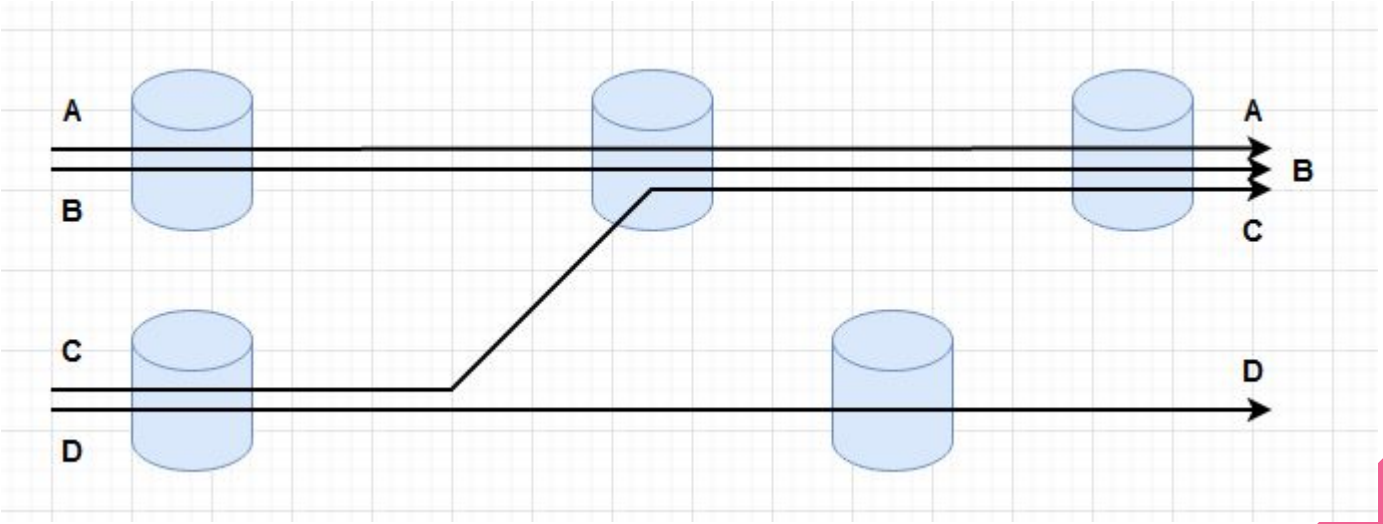
Bandwidth = 100



Extra exercise

Bandwidth = 100

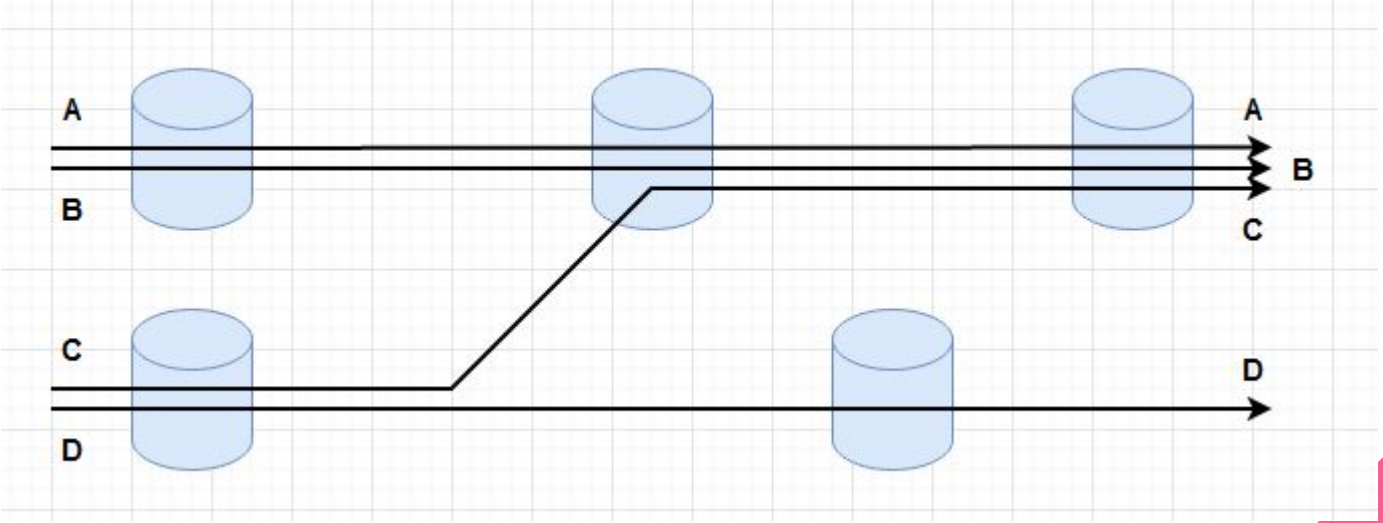
- Start with all flows at rate 0
- Increase the flows until there is a new bottleneck in the network
- Hold fixed the rate of the flows that are bottlenecked
- Go to step 2 for any remaining flows



Extra exercise

Bandwidth = 100

- Start with all flows at rate 0
- Increase the flows until there is a new bottleneck in the network
- Hold fixed the rate of the flows that are bottlenecked
- Go to step 2 for any remaining flows

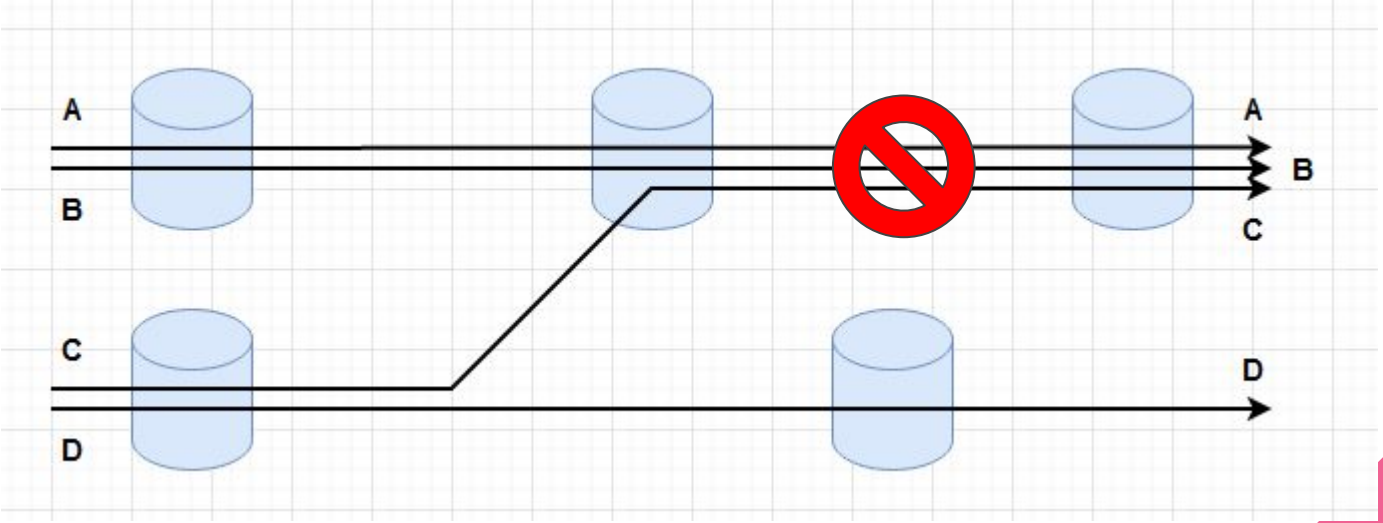


A	0
B	0
C	0
D	0

Extra exercise

Bandwidth = 100

- Start with all flows at rate 0
- Increase the flows until there is a new bottleneck in the network
- Hold fixed the rate of the flows that are bottlenecked
- Go to step 2 for any remaining flows

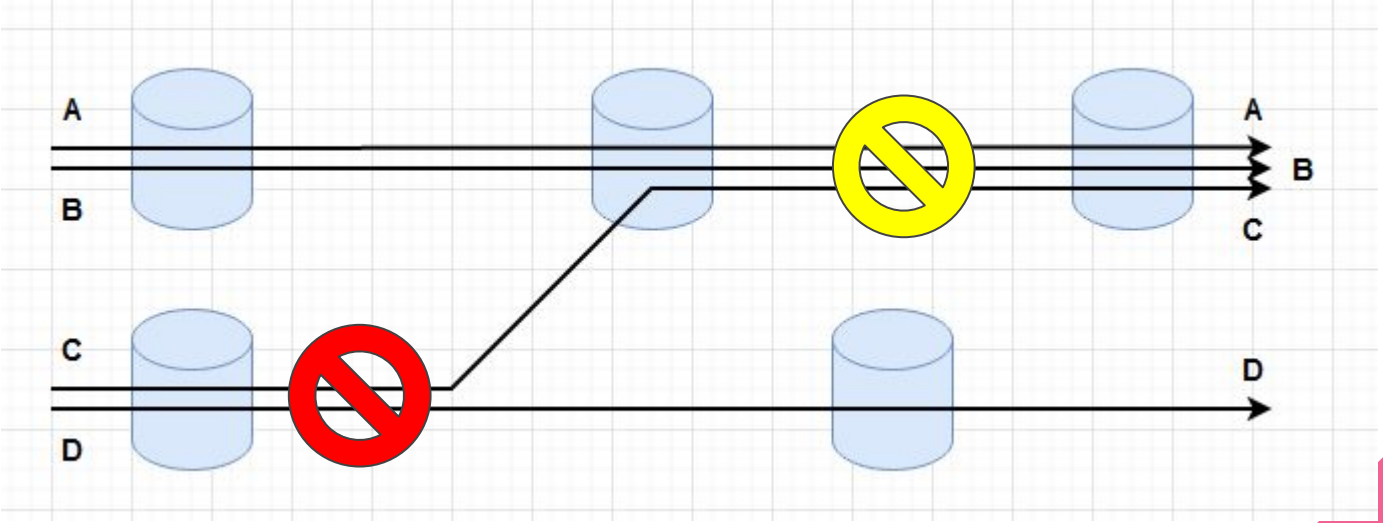


A	33
B	33
C	33
D	0

Extra exercise

Bandwidth = 100

- Start with all flows at rate 0
- Increase the flows until there is a new bottleneck in the network
- Hold fixed the rate of the flows that are bottlenecked
- Go to step 2 for any remaining flows



A	33
B	33
C	33
D	67

Q&A, Extra OH