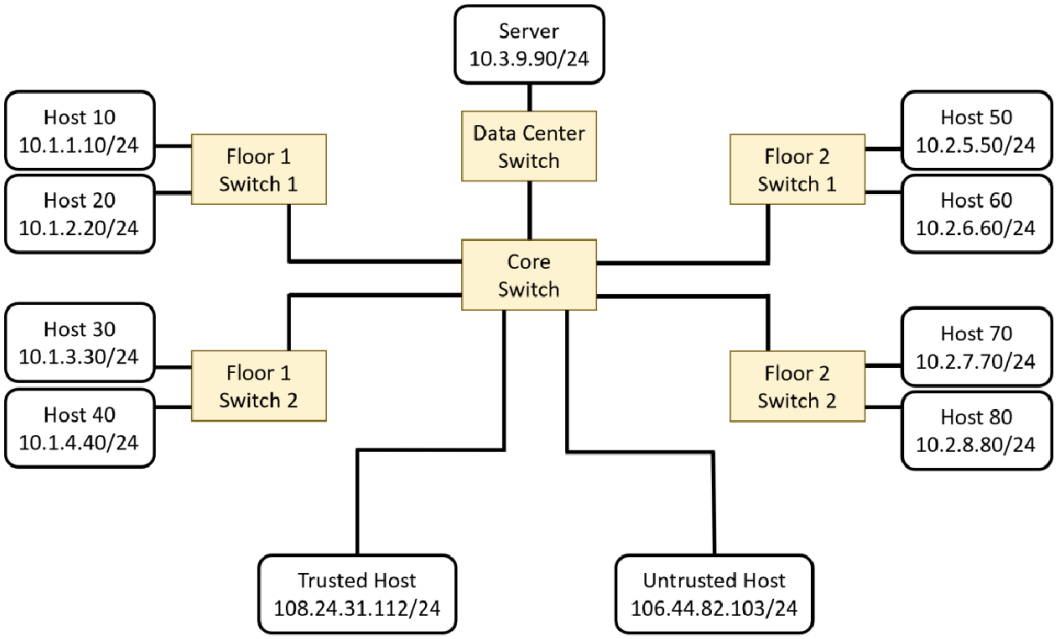
Rohan Kapur
CSE 150 Spring 2023
Final Project
Due 6/10/23

The following network topology is implemented for this project:



With the network traffic rules being implemented such that **all hosts are able to communicate, EXCEPT:**

- **Untrusted Host cannot send ICMP traffic to Host 10 to 80, or the Server.**
- **Untrusted Host cannot send any IP traffic to the Server.**
- **Trusted Host cannot send ICMP traffic to Host 50 to 80 in Department B, or the Server.**
- **Trusted Host cannot send any IP traffic to the Server.**
- **Hosts in Department A (Host 10 to 40) cannot send any ICMP traffic to the hosts in Department B (Host 50 to 80), and vice versa.**

My controller implementation regulates network traffic by obtaining the destination IP address for the received packet if it's an IP packet (using "packet.find('ipv4').dstip") and then based on the switch from which the packet was sent to the controller ("switch_id" argument for the "do_final" method) and the port the packet was received on ("packet_on_switch" argument for the "do_final" method), the packet is sent to the correct output port using a long conditional chain that constructs and installs a flow table entry on the switch for the the packet and similar packets as follows:

- For packets received from the data center switch that only the server is connected to:
    - Packets coming from the core switch (port 1) are sent out to the server (port 2)
    - All other packets are sent out to the core switch (port 1)

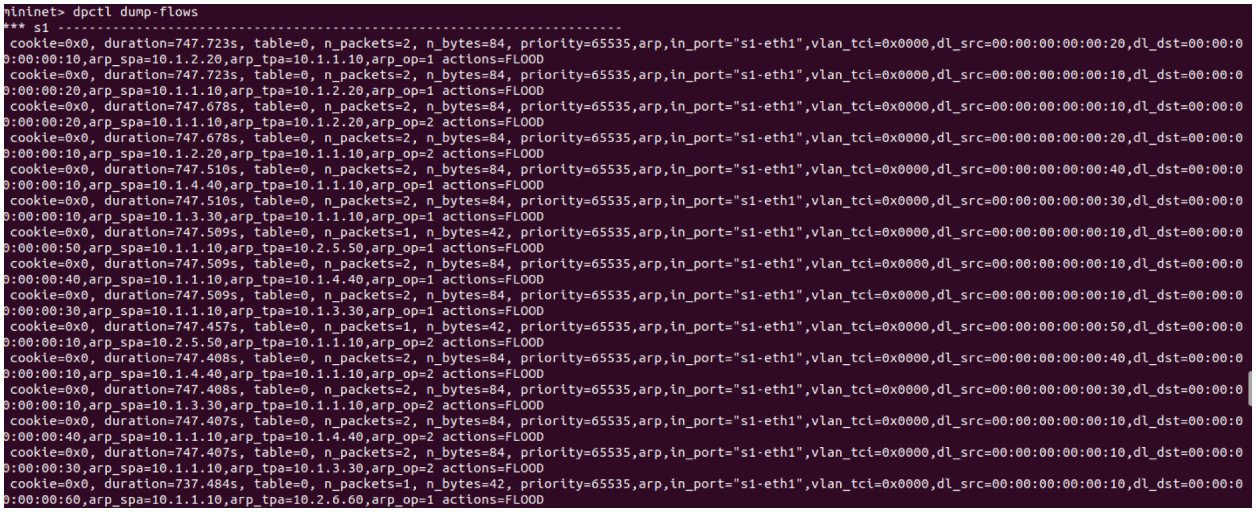
- For packets received from the core switch:
    - Packets sent from the data center switch (port 1) are all forwarded to their destination switch ports
    - Packets sent from the floor 1 switches (ports 2 or 3) are forwarded to their destination ports *unless* they are for floor 2 switches (ports 4 or 5) in which case they're dropped and vice-versa
    - Packets sent from the trusted host (port 6) is forwarded to the destination port *unless* they are meant for the server in which case they are dropped
    - Packets sent from the untrusted host (port 7) are all dropped unless they're meant for the trusted host (port 6)
- Packets from other switches are kept within the switch if the destination is on the same switch or sent out the core switch port (port 1 on all non-core switches) otherwise

Running "pingall" with these rules and implementation yields the following output:

```
mininet> pingall
*** Ping: testing ping reachability
h10 -> h20 h30 h40 X X X X X server trusted
h20 -> h10 h30 h40 X X X X X server trusted
h30 -> h10 h20 h40 X X X X X server trusted
h40 -> h10 h20 h30 X X X X X server trusted
h50 -> X X X X h60 h70 h80 X server X
h60 -> X X X X h50 h70 h80 X server X
h70 -> X X X X h50 h60 h80 X server X
h80 -> X X X X h50 h60 h70 X server X
hacker -> X X X X X X X X X trusted
server -> h10 h20 h30 h40 h50 h60 h70 h80 X X
trusted -> h10 h20 h30 h40 X X X X hacker X
*** Results: 54% dropped (50/110 received)
```

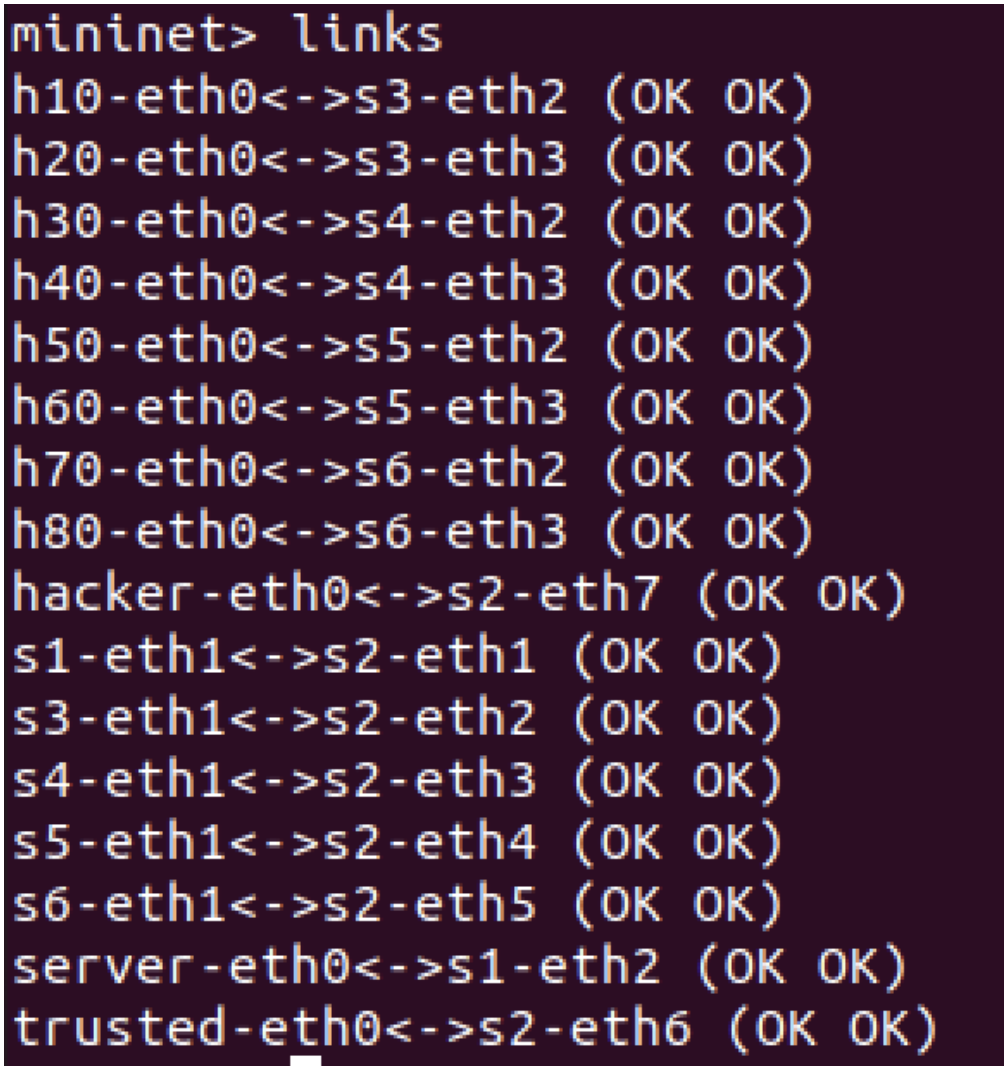
with **54%** of the packets dropped, which seems to be correct in following the packet routing and blocking specified in the project instructions.

The "dpctl dump-flows" command, which outputs the flow-table entries installed on each switch that are used to transmit the packets received on the switches without having to send them to the controller every time for processing, is *very* long, but part of it is shown here for the server switch (switch 1):



This allows for similar packets to be regulated within the switches rather than having to be sent to the controller every time for processing.

The “links” command outputs the following, which shows the correct implementation for the network topology:



S1 is the server switch, s2 is the core switch, s3 is floor 1 switch 1, s4 is floor 1 switch 2, s5 is floor 2 switch 1, s6 is floor 2 switch 2, s7 is floor 2 switch 2, s6 trusted host