Final Project: Implementing a Simple Router

In the previous lab you implemented a simple firewall that allowed ARP and TCP packets, but blocked all other packets. For your final project, you will be expanding on this to implement routing between devices on different subnets and implementing firewalls for certain subnets. The idea is to simulate an actual production network. You will be using ideas from Lab 1 to help construct the mininet topology, and ideas from Lab 3 to implement the rules allowing for traffic to flow through your network. Please refer back to those Labs for guidance on how to complete this assignment.

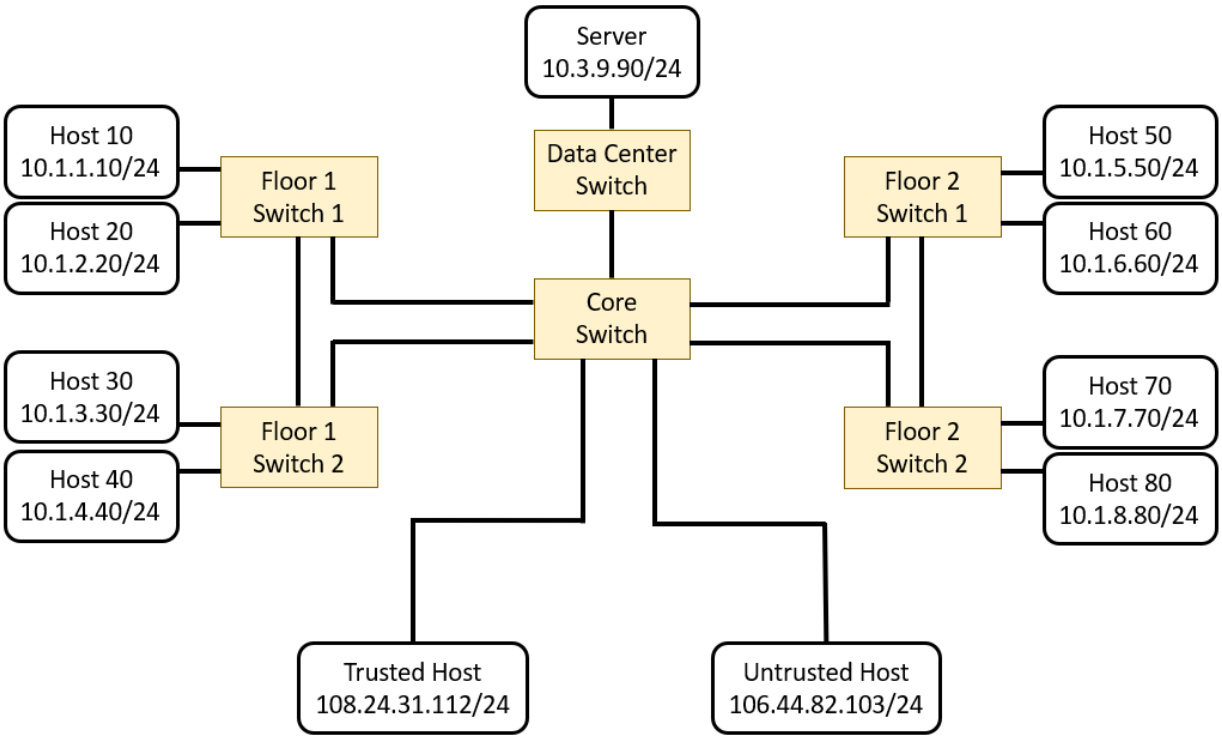
Assignment:

For this lab, we will be constructing a network for a small company. The company has a 2-floor building, with each floor having its own switches and subnets. Additionally, we have a switch and subnet for all the servers in the data center, and a core switch connecting everything together.

Your device's roles and IP addresses are as follows:

Device	Mininet Name	IP Address	Description
Floor 1 Hosts	h10, h20, h30, h40	10.1.1.10/24 10.1.2.20/24 10.1.3.30/24 10.1.4.40/24	Computers on floor 1 of the Department A in the company.
Floor 2 Hosts	h50, h60, h70, h80	10.2.5.50/24 10.2.6.60/24 10.2.7.70/24 10.2.8.80/24	Computers on floor 2 of the Department B in the company.
Trusted Host	h_trust	108.24.31.112/24	A trusted computer outside our network. This host is owned by certified employee from Department A.
Untrusted Host	h_untrust	106.44.82.103/24	An untrusted computer outside our network. We treat this computer as a potential hacker.
Server	h_server	10.3.9.90/24	A server used by our internal or trusted hosts.

The topology will look as follows:



Your goal will be to allow or block traffic between the hosts and servers. In this assignment, you will be allowed (and encouraged) to flood all non-IP traffic in the same method that you did in Lab 3 (using a destination port of of.OFPP_FLOOD). **However, you will need to specify specific ports for all IP traffic.** You may do this however you choose-- however, you may find it easiest to determine the correct destination port by using the destination IP address and source IP address, as well as the source port on the switch that the packet originated from. Additional information has been given to you in the do_final() function to allow you to make these decisions. Please see the comments in the provided code for guidance.

Requirements:
To protect our servers from the untrusted Internet, we will be blocking all IP traffic from the Untrusted Host to the Server. To block the Internet from discovering our internal IP addresses, we will also block all ICMP traffic from the Untrusted Host to anywhere internally (i.e., Host 10-80 and the Server). For the trusted host, it can send any traffic to the hosts in the Department A (Host 10, 20, 30, 40). Meanwhile, similar as the untrusted host, the trusted host cannot send any ICMP and IP traffic to the server, and it cannot send ICMP traffic to the hosts in the Department B (Host 50, 60, 70, 80).
Additionally, all ICMP traffic from the hosts in Department A (Host 10, 20, 30, 40) to the hosts in Department B (Host 50, 60, 70, 80) should be blocked and vice versa.

In your report, you will need to explain how you implemented the various requirements and show that they work properly with screenshots. If you need help figuring out how to do this, look back to previous assignments and see how you tested them.

Provided Code:

Available in a ZIP file here.
https://users.soe.ucsc.edu/~qian/code/final_project.zip

We have provided you with starter code (skeleton files) to get you started on this assignment. The controller file (final_controller_skel.py) needs to be placed in ~/pox/pox/misc, and the mininet file (final.py) should be placed in your home directory (~). This time, you will need to modify both files to meet the lab requirements.

You will be using slightly different commands to create the Hosts and Links in the Mininet file to give you more information to make decisions within the Controller file. Additionally, you will notice that you have additional information provided in the do_final function. This is documented in the comments within the files.

Summary of Goals:

- Create a Mininet Topology (See Lab 1 for help) to represent the above topology.
- Create a Pox controller (See Lab 3 for help) with the following features:
    - 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.

Testing:

You may test with mininet commands and observing packets with Wireshark inside your VM. Please have a detailed illustration (with screenshots) about how you tested your codes in the report.

Grading Rubric:

Total: 100 points
Submit both the code and PDF report.

20 points: Mininet Topology (use proper mininet commands and screenshots to justify in your report)
- 10: Devices are successfully created.
- 10: Links are successfully created, and the topology is correct.
- 10: IP addresses are correct.

60 points: Pox Controller (use proper mininet commands and screenshots to justify in your report)
- 20: Hosts can communicate.
    - 10 point deduction if rules not installed in flow table.
    - 10 point deduction if IP traffic is implemented using OFPP_FLOOD.
- 10: Untrusted Host cannot send ICMP traffic to Host 10 to 80
    - 5 point deduction if Untrusted Host cannot send ANY traffic to the hosts.
- 10: Untrusted/Trust Host cannot send any traffic to Server
- 10: Trusted Host cannot send ICMP traffic to Host 50 to 80
    - 5 point deduction if Trusted Host cannot send ICMP traffic to Host 10 to 40
- 10: Host 10 to 40 cannot send ICMP traffic to Host 50 to 80

20 points: Quality of the report.
Credits are given based on the clarity of your illustration and result justification. You must include screenshots proving your code works. **These screenshots must come from your own code.** This will be tested and submitting screenshots of someone else's code can be considered an academic integrity violation.
Note that for the report, we will **not** be telling you what commands to run to verify the assignment goals are met. Figuring out how to prove your work is a part of this assignment. A good report should comprehensively analyze the correctness of your implementation and present the results in a clear way. You could follow the "Summary of Goals" or the "Grading Rubric" to decide how to demonstrate your implementation in the report.

Partial credit may be awarded for incomplete assignments based upon the submitted code and explanations in report as to why something may not be functioning properly. If you are not able to get the expected results, below your screenshot, explain what you think is going on (for partial credit).

Deliverables:

1. **project.pdf**: See instructions above.
2. **final_skel.py**: Your topology code.
3. **finalcontroller_skel.py**: Your controller code.
4. **README.txt**: A readme file explaining your submission.