# CE 150/L Final Project: Implementing Policy-Based Routing

In Lab 3 you implemented a simple firewall that allowed ARP and TCP packets to be forwarded, but blocked all other packets. For your final project, you will be expanding on this to implement policy-based routing between devices on different subnets. The idea is to simulate an actual production network. You will be using what you learned in Lab 1 to help construct the mininet topology, and what you learned in 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. Note that this is an individual project.

## Assignment:

For this lab, you will construct a network that consists of 4 subnets (10.0.1.0/24, 10.0.2.0/24, 10.0.3.0/24, 10.0.4.0/24). Each subnet has one switch that is connected to a core switch as shown in Figure 1 below. The devices' addresses are as follows:

| | Subnet | Mininet Name |
|---|---|---|
| Host 1 | 10.0.1.1/24 | h1 |
| Host 2 | 10.0.1.2/24 | h2 |
| Host 3 | 10.0.2.1/24 | h3 |
| Host 4 | 10.0.2.2/24 | h4 |
| Host 5 | 10.0.3.1/24 | h5 |
| Host 6 | 10.0.3.2/24 | h6 |
| Host 7 | 10.0.4.1/24 | h7 |

Your first goal will be to allow traffic to be transmitted between all the hosts from the same subnet. In this assignment, you will need to specify specific ports for all IP traffic. Although you can accomplish this in different ways, you may find it easier to determine the correct destination port by using the destination IP and source IP address, as well as the source port on the switch that the packet originated from.

Your second goal is to only allow traffic between subnet 10.0.1.0/24 and 10.0.3.0/24. Any traffic from other subnets should be blocked. For example, host 10.0.1.1 should be able to communicate with host 10.0.3.1 and vice versa. However, host 10.0.4.1 can't communicate with host 10.0.1.1 or host 10.0.3.1.

Your third goal is similar to your second goal. The goal is to allow traffic between subnet 10.0.2.0/24 and 10.0.4.0/24. Any traffic from other subnets should be blocked.

As an extra credit, your fourth goal is not to allow the flood of all ARP traffic similar to that you did in Lab 3. You are only allowed to flood packets when you have to. For example, packet with destination MAC address ff:ff:ff:ff:ff:ff should be flooded. 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. The routing policies that you will need to implement are described in the table below.

| Policy | Description |
|---|---|
| 1 | Allow hosts from each subnet to communicate with hosts within the same subnet. |
| 2 | Allow hosts on subnet 10.0.1.0/24 can communicate ONLY with hosts on subnet 10.0.3.0/24 and vice versa.<br><br>Any other traffic should be dropped. |
| | Allow hosts on subnet 10.0.2.0/24 can communicate ONLY with hosts on subnet 10.0.4.0/24 and vice versa.<br><br>Any other traffic should be dropped. |
| Extra Credit: | Instead of flooding ARP packets like in Lab 3, create a rule that help switches know the mac addresses of all the hosts in the network. This can help improve the performance of our network since we will only broadcast when we have to.<br><br>Remember, you are only allowed to flood ARP packets that are meant to be broadcasted such as ARP request. |

The topology you need to emulate looks as follows:

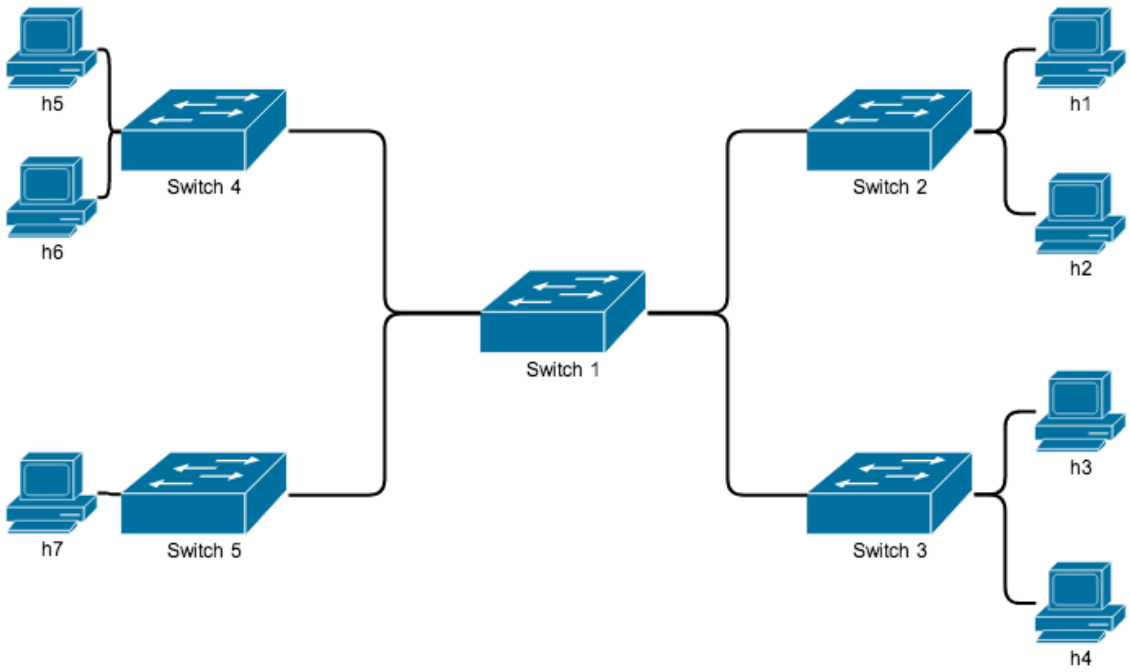


**Figure 1**

## Provided Code:

Available in a ZIP file here.

We have provided you with starter code (skeleton files) to get you started on this assignment. The controller file (finalcontroller_skel.py) needs to be placed in ~/pox/pox/misc, and the mininet file (final_skel.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:

1. Create a Mininet Topology (see Lab 1 for help) to represent the above topology.
2. Create a Pox controller (see Lab 3 for help) with the following features:
    2.1. All hosts from the same subnet can communicate with other hosts within the same subnet.
    2.2. Hosts on subnet 10.0.1.0/24 can communicate ONLY with hosts on subnet 10.0.3.0/24 and vice versa
    2.3. Hosts on subnet 10.0.2.0/24 can communicate ONLY with hosts on subnet 10.0.4.0/24 and vice versa
    2.4. Extra credit: flood ARP packets that are meant to be broadcasted such as ARP request.

## Testing:

You may test with ping commands, iperf, xterm windows, and observing packets with Wireshark inside your VM.

## Demo:

You will be required to demonstrate your project to the TAs. Note that demos are mandatory. You will need to explain how you implemented the various requirements and show that they work properly. If you need help figuring out how to do this, look back to previous assignments and see how you tested them.

Please have a plan in place to show you have implemented all the requirements during your demonstration to the TAs.

## Deliverables:

- Source code
- Readme documenting your code
- Demo

## Grading Rubric:

Total: 100 points

30 points: Mininet Topology
    10: Devices successfully created.

10: Links successfully created.
10: IP addresses correct.

50 points: Pox Controller

20: All hosts can communicate with other hosts from the same subnet.
15: Hosts on subnet 10.0.1.0/24 can communicate ONLY with hosts on subnet 10.0.3.0/24 and vice versa
15: Hosts on subnet 10.0.2.0/24 can communicate ONLY with hosts on subnet 10.0.4.0/24 and vice versa
10 Extra credit: flood ARP packets that are meant to be broadcasted such as ARP request.
10 point deduction if rules not installed in flow table.
10 point deduction for each violation of the rules.

20 points: Demonstration

Note that for the demo, 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.

Partial credit may be awarded for incomplete assignments based upon demonstration and explanations as to why something may not be functioning properly.

## Submission and Due Date:

The project can be demo'ed during the last week of class (week of 12.04) during a lab session. There will be two addition demo opportunities during finals week, namely Monday (12.11) between 11-1pm and Tuesday (12.12) between 4-6pm. Your code and README file must be submitted by 12.12.