Lab 6: Designing and implementing a small network

50.012 Networks

Hand-out: December 3
eDimension hand-in: December 15, 11:59pm

1 Learning Objectives

- Use knowledge from the class to design and implement a small network
- Your mission is to connect a set of virtual hosts in mininet correctly through switches and routers, and to configure some key services for them

2 Setup

- This exercise again assumes that you have a running mininet installation
- Download the lab6.zip from eDimension and unpack to some local folder
  - Tip: Don't use space in folder/file names on Linux, might break scripts
- Change directory into that folder and execute the `install.sh` script by typing:

  ```
  sudo bash ./install.sh
  ```

- It will install `dnsmasq`, a useful lightweight tool that provides DNS and DHCP services for constructing small networks, see more at `http://www.thekelleys.org.uk/dnsmasq/doc.html`

3 Warming Up

The general setup in this lab is shown in Figure 1. 1 gateway, 2 local servers and 5 local hosts are connected via 2 switches. The gateway is also connected to the *Internet*, in our case the part shown as remote AS.

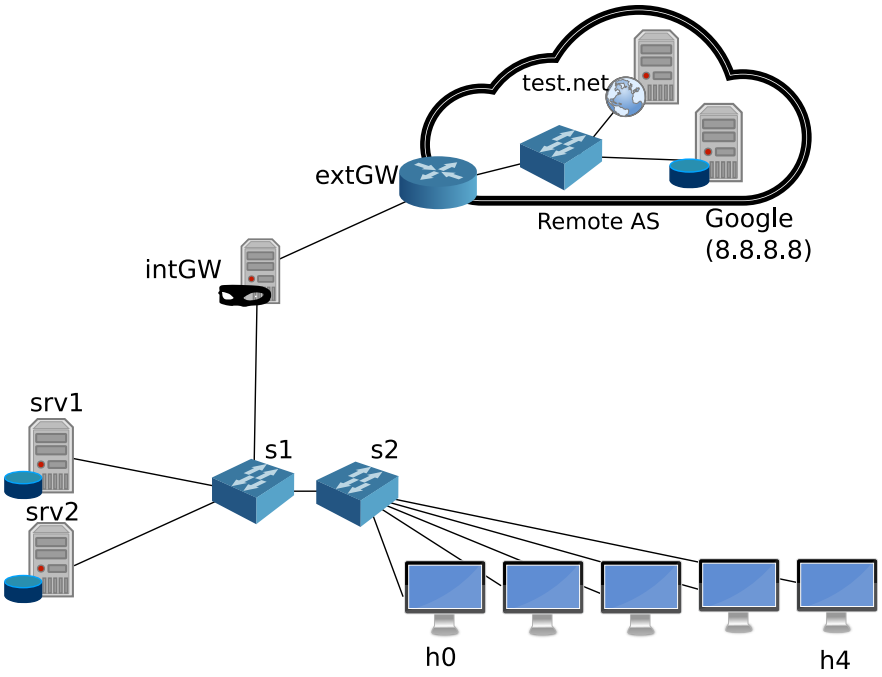

Figure 1: Basic topology in Mininet

- Start up mininet

  ```
  sudo python ./net.py
  ```

- Open an xterm on `h1` and check the local network configuration using:

  ```
  mininet> xterm h1
  # ifconfig
  # route -n
  ```

- Q1: What is the IP subnet that is chosen for the hosts?
- Q2: Are the two servers `srv1` and `srv2` in the same subnet?
- Q3: Test if you can observe the switch with `tracepath` from `h1` to `srv1`? Hint: use the `-n` option for `tracepath` to prevent unnecessary DNS lookups. Why are you not able to observe the switch?
- Q4: What is the gateway for the devices `srv1`, `srv2`, and the hosts `h0` to `h4`?
- Q5: Can you ping/reach the server test.net (8.8.8.2) from `h1`? If not, do you have an idea what goes wrong?
- Q6: Is a DHCP server running in the local network? On which machine? You can use `dhclient <IFNAME>` (where `IFNAME` correspond to the name of a network interface) to request a new IP address manually:

  ```
  mininet> xterm h1
  # dhclient h1-eth0
  ```

- Observe the DHCP traffic using Wireshark if necessary. Note that Wireshark is dissecting it as DHCP but in the packet header you might find Bootstrap Protocol (BOOTP).

4 Configuration of the System

4.1 Changing the DHCP configuration

Open the DHCP server configuration file at `srv1DHCP.conf` using your favorite editor. Look at the settings and try to understand what they mean.

- Q7: Do you find something that might need to be improved?

Do that change, save the file, and restart the `net.py` mininet simulation

- Q8: In the open mininet session, open an xterm on h1 again and ping Google (8.8.8.8). Can you reach it now?

4.2 DNS

Let us now try to configure `h1` to use our custom DNS server. Note: this can be a bit tricky. For best results, start the mininet session, and then

- `sudo service network-manager stop` on your host machine (you will lose your connection to the Internet)
- `sudo nano /etc/resolv.conf` and replace the `127.0.0.1` IP with `8.8.8.8`
- Now it should work until you restart network manager (with `sudo service network-manager start`) or you restart mininet
- Q9: On the `h1` host, ping test.net. Can you reach it? Why? Try using `dig` or `nslookup` to find out more. What is the IP of test.net?

4.3 Observing NAT in action

- Q10: In the provided setup, one node provides NAT for the hosts with private IP address. Which node is this?

Use wireshark on that host to inspect incoming and outgoing connections Have a look at the `net.py` script to see what is going on in the `enableNAT()` function. This is all it needs to configure a host to do NAT (under Linux).

4.4 Simple Firewalling

The NAT was actually set up using `iptables`, which can be used as Linux firewall application In a nutshell, a firewall can prevent or allow incoming/outgoing connections to/from a machine.

- In particular, specific rules can be added to **drop** (discard) traffic from certain sources, or using certain protocols or ports

Open an xterm on `intGW` and add a rule to block all traffic from srv2 specifically.

- Q11: What is the rule you added? Test if it works, i.e. if you can still ping 8.8.8.8 from srv2 after the rule is effective. Ideally, you should not!

Use either the Internet, or the manpage, or the net.py script as reference. In `net.py`, a similar rule is used to emulate un-routable private IP-addresses.

- A good example tutorial is found at
  `http://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/`
- *Hint1*: there are different arguments like `-I` and `-A`. They change the order in which rules are evaluated. `-I` puts the rule you insert to the front.
- *Hint2*: there are different *chains* like `INPUT` and `FORWARD`. The input chain applies to all packets direct to the host, while the forwarding chain applies to all packets forwarded by the host (e.g. on a router or NAT host).
- *Hint3*: there are different tables and the one that we are using for this exercise is the (default) `filter` table that has `INPUT`, `OUTPUT`, and `FORWARD` chains.
- *Hint4*: there are different *commands* like `DROP`, `REJECT` and `ACCEPT`. The main difference between `DROP` and `REJECT` is that in the latter case, an error message is sent to the source.

5 What to Hand in

Please provide a writeup that includes your group's answers to Q1 - Q11.