Project #5: Set up your own IP network

CS 352 Internet Technology

Released: April 10th, 2022; Due: April 29th, 2022

Instructions

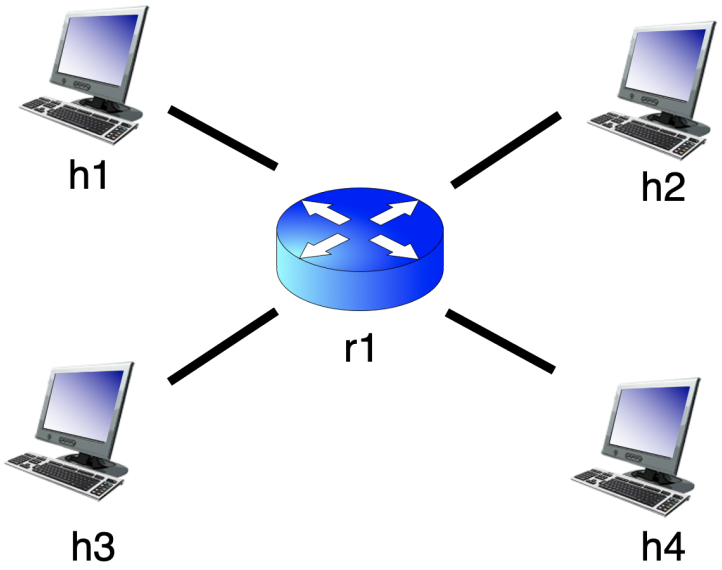
Please read these instructions carefully before you begin.

1. You must work on this project in teams of 2.
2. You are free to discuss the project on Piazza or through other means with your peers and the instructors. You may refer to the course materials, textbook, and resources on the Internet for a deeper understanding of the topics. However, you cannot lift solutions from other students or from the web including GitHub, Stack Overflow, or other resources. Do not post this project to question-answering services like Chegg. All written and programmed solutions must be your team's original work. We run sophisticated software to detect plagiarism and carefully monitor student answers. If you are in doubt, please ask us.
3. You cannot post your solutions to this project on your personal GitHub page or on other web services hosting class materials.
4. For each question in the project report, please be clear and concise. Vague and rambling answers will receive zero credit.
5. For the report question on collaboration, please include anyone you discussed the project with, and also any resources you consulted to learn how to solve this project, including URLs of pages visited on the Internet. Please be specific about the aspect of the project that you got help with. You must be thorough and as complete as possible here. It is mandatory to answer this question.
6. We encourage you to *start early* and get the bulk of the work for this project done the week(s) before it is due, rather than keeping it until the submission date.
7. If you have any questions or clarifications on the project, please post them on Piazza or contact the course staff. We are here to help.

Overview

In this project, you will set up a small IP network consisting of hosts and a router. Your network will function with full end-to-end connectivity, and you will experiment with tools like `ping` and `traceroute`.

Consider the following network topology:



Here, `r1` is an IP router and `h1 ... h4` are endpoints. In a safe experimental setting running inside a virtual machine, you will use the `ip` suite of commands to set up the right IP addresses for all the interfaces, and the right routing table entries for all the routers and endpoints in this network. Welcome to the AS352 network!

Step 1: Set up mininet

To create a safe virtualized environment, we will use a tool called mininet, which uses linux containers (see `https://en.wikipedia.org/wiki/Linux_namespaces`) to allow you to create a small network of hosts and routers that can run inside your laptop. (If you're interested in learning more about mininet, you can read about it from the paper entitled "A network in a laptop: Rapid prototyping for software-defined networks," found at `http://conferences.sigcomm.org/hotnets/2010/papers/a19-lantz.pdf`.)

First, download the standard mininet virtual machine (note, this is a 500+ MByte file).

`https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-i386.zip`

Set up the VM and play around according to the instructions at the link below. We recommend that you use VirtualBox. By the end of this step, you must be able to log into the VM and run basic

terminal commands like `ls`.

`http://mininet.org/vm-setup-notes/`

Step 2: Play with the topology Python file

Within the virtual machine, run the attached network topology file, `AS352.py`, which invokes mininet's Python API to create a network topology with 4 endpoints (hosts) and 1 router. Note the use of `sudo` while running the program. Note that this program will be unsuccessful on any machine without a viable mininet installation. In particular, you cannot run `AS352.py` on ilab.

```
mininet@mininet-vm:~/project5$ sudo python AS352.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 r1
*** Adding switches:

*** Adding links:
(h1, r1) (h2, r1) (h3, r1) (h4, r1)
*** Configuring hosts
h1 h2 h3 h4 r1
*** Starting controller
c0
*** Starting 0 switches

*** Routing Table on Router:
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
*** Starting CLI:
mininet>
```

Within the mininet shell, you can do various things to control the hosts and router we just instantiated. One highly convenient aspect of mininet is that the hosts, the router, and the VM all share the same filesystem, so the same files are visible from all hosts and the router. Also, you can run commands, dump outputs into a file, and access it from any host or router or a regular terminal inside the VM. Let's walk through a few examples of playing around with the mininet topology.

Step 2.1. Run the `ls` command on host h1.

```
mininet> h1 ls
AS352.py
mininet>
```

Note that the command is prefixed by `h1` to indicate to mininet that the command must be run on host `h1`. You can replace `ls` by any command which can run on `h1`, by just typing

```
h1 <command>
```

where `<command>` is any command that would run on a regular Linux terminal. You can run `ls` on any host or router by changing the prefix of the command:

```
<host or router name> ls
```

Step 2.2. Inspect the network topology from within mininet:

```
mininet> net
h1 h1-eth0:r1-eth1
h2 h2-eth0:r1-eth2
h3 h3-eth0:r1-eth3
h4 h4-eth0:r1-eth4
r1 r1-eth1:h1-eth0 r1-eth2:h2-eth0 r1-eth3:h3-eth0 r1-eth4:h4-eth0
c0
mininet>
```

Note that `net` is not prefixed by any host or router name. It is a native mininet command.

Step 2.3. You can even run commands on the router! Mininet allows you to treat both endpoints and the router as standard Linux machines, on which you get to run "regular" Linux commands. Let's dump the routing table on `r1`.

```
mininet> r1 ip route
mininet>
```

By default, there is nothing in the routing table on `r1`. As you may have guessed, that is one of the things that you will add in this project.

Step 2.4. Mininet can also read a list of commands from a file. Suppose we added the text

```
h1 ls
net
r1 ip route
```

into the file `commands.txt`. Then, you can use the `source` mininet command to run all the commands in sequence, one after another. For example:

```
mininet> source commands.txt
AS352.py  commands.txt
h1 h1-eth0:r1-eth1
h2 h2-eth0:r1-eth2
h3 h3-eth0:r1-eth3
h4 h4-eth0:r1-eth4
r1 r1-eth1:h1-eth0 r1-eth2:h2-eth0 r1-eth3:h3-eth0 r1-eth4:h4-eth0
c0
mininet>
```

You can find more examples of mininet commands and usage at `http://mininet.org/walkthrough/#interact-with-hosts-and-switches`

Step 3: Set up your network

Now we get to the crux of the project. You need to set up the network interfaces on all hosts and routers, as well as the routing tables, so that end to end IP communication is possible between all hosts and the router interfaces. You will do this in three steps.

Step 3.1. Set up the IP addresses of all interfaces. Please assign the following IP addresses to the corresponding network interface below.

10.0.0.1 h1-eth0
192.168.0.1 h2-eth0
10.0.0.3 h3-eth0
192.168.0.3 h4-eth0
10.0.0.2 r1-eth1
192.168.0.2 r1-eth2
10.0.0.4 r1-eth3
192.168.0.4 r1-eth4

Use the `ip addr` command to set up the correct IP address for each interface on each machine. You need to set up 8 interface addresses in total. If you need help in using the command, try running `ip addr help`.

Step 3.2. Set up the default routes for the hosts. The hosts `h1 ... h4` are only connected to the rest of the network through one interface. Any external communication must hence use a *default* route, which moves the packets on the host towards the gateway router, `r1`. For example, all traffic on host h1 must use the interface/device `h1-eth0` to reach any destination (other than itself).

Use the command `ip route` command to set up all the routes. If you need help understanding the command, try running `ip route help`. You will set up 4 default routes, one on each endpoint `h1 ... h4`.

Step 3.3. Set up the routes on the router. We need to configure the router to interconnect the endpoints by matching incoming packets with the correct outgoing network interface. The forwarding table entries on router `r1` must look like:

```
Destination IP address -> Output port
--------------------------------------
h1 -> r1-eth1
h2 -> r1-eth2
h3 -> r1-eth3
h4 -> r1-eth4
```

Once again, use the `ip route` command to set up these routes. You will need to set up 4 routes, one for each destination endpoint.

Step 3.4. Test your network: You've done all the hard work. Now check that your network works as intended, using the `ping` and `traceroute` commands.

A correctly configured network must be able to execute a successful `ping` from any host or the router to any valid IP address inside the network.

For example, you can ping `h4` from `h1`, to check reachability:

```
mininet> h1 ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3) 56(84) bytes of data.
64 bytes from 192.168.0.3: icmp_seq=1 ttl=63 time=168 ms
64 bytes from 192.168.0.3: icmp_seq=2 ttl=63 time=0.046 ms
64 bytes from 192.168.0.3: icmp_seq=3 ttl=63 time=0.044 ms
^C
--- 192.168.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.044/56.315/168.855/79.577 ms
mininet>
```

You can also run `traceroute` between any two points. For example, a traceroute from `h1` to `h4` would show something like this:

```
mininet> h1 traceroute 192.168.0.3
traceroute to 192.168.0.3 (192.168.0.3), 30 hops max, 60 byte packets
 1  10.0.0.2 (10.0.0.2)  0.018 ms  0.003 ms  0.003 ms
 2  192.168.0.3 (192.168.0.3)  0.008 ms  0.003 ms  0.004 ms
mininet>
```

Note that there are 2 hops in the traceroute output: the router and the destination of the traceroute, with the IP address of the router interface connected to `h1` showing up on the first row.

In case you received an error that the `traceroute` command isn't found, install `traceroute` by running

```
sudo apt-get update && sudo apt-get install traceroute
```

How we will test your commands

Once you have set up all the interfaces and routes, put all your commands into a file called `commands.txt`. Please do not modify `AS352.py`; your commands must work with the unmodified Python script.

When all the commands in your commands file are run one after another (by typing `source commands.txt`), subsequent commands should show (1) correct configuration of the interfaces and routes (demonstrated through `ip addr` or `ifconfig` and `ip route`) and (2) full connectivity between all the nodes in the network, such that a `ping` or `traceroute` from any endpoint or router to any valid IP address in the network should succeed. We may test connectivity between any or all pairs of endpoints and the router. We will only use `ping` and `traceroute` to test connectivity.

You are free to use tools like `iperf` to do more sophisticated testing. An example of a test using `ping` might look like:

```
*** Starting CLI:
mininet> source commands.txt
mininet> h1 ping 192.168.0.4
PING 192.168.0.4 (192.168.0.4) 56(84) bytes of data.
64 bytes from 192.168.0.4: icmp_seq=1 ttl=63 time=1070 ms
64 bytes from 192.168.0.4: icmp_seq=2 ttl=63 time=60.8 ms
64 bytes from 192.168.0.4: icmp_seq=3 ttl=63 time=0.030 ms
64 bytes from 192.168.0.4: icmp_seq=4 ttl=63 time=0.047 ms
^C
--- 192.168.0.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3010ms
rtt min/avg/max/mdev = 0.030/282.820/1070.311/455.336 ms, pipe 2
mininet>
```

Please ensure that mininet does not crash when running your `commands.txt`.

Notes that may be useful

Mininet has an FAQ/help page at `https://github.com/mininet/mininet/wiki/FAQ`. There is also a mailing list where you can post (after checking the FAQ and the archives) if you have questions.

To reduce the file size of the disk image, the mininet VM does not come with a pre-installed graphical interface. Hence, using the VM may require you to get comfortable with the Linux terminal/command line. You may find it convenient to use multiple terminals either from the VM console or over SSH using terminal multiplexers like `tmux` and `screen`. Alternatively, you may try following the instructions at

`https://github.com/mininet/openflow-tutorial/wiki/Set-up-Virtual-Machine#alternative-run-a-gui-in-the-vm-console-window`

to install a graphical interface on the mininet VM. We have not tested those instructions ourselves, however.

You may be surprised to learn that even endpoints have routing tables. When endpoints are connected through multiple network interfaces, a forwarding table on the endpoint determines through which interface a given packet is transmitted.

You will often need to "debug" your network configuration by checking whether packets reached the correct interface with the correct header values. Use a packet sniffer to do this. You may be familiar with wireshark, but it requires a graphical user interface. Instead, we recommend using `tcpdump` which is a good command-line equivalent. For example, the following command uses tcpdump to sniff the `r1-eth2` interface on `r1`, and dumps the sniffed packets to a file `r1-dump.txt`.

```
mininet> r1 tcpdump -v -U -i r1-eth2 -x > r1-dump.txt &
mininet> h1 ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=63 time=0.021 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=63 time=0.028 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=63 time=0.037 ms
^C
--- 192.168.0.2 ping statistics ---
3 packets transmitted, 3 received, 0\% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.021/0.028/0.037/0.009 ms
mininet>
```

In a separate window, you can check sample output of the `tcpdump`:

```
mininet@mininet-vm:~/project5$ head r1-dump.txt
18:31:29.326269 ARP, Ethernet (len 6), IPv4 (len 4),
  Request who-has 192.168.0.2 tell 192.168.0.1, length 28
        0x0000:  0001 0800 0604 0001 feb5 2946 9994 c0a8
        0x0010:  0001 0000 0000 0000 c0a8 0002
18:31:29.326278 ARP, Ethernet (len 6), IPv4 (len 4),
  Reply 192.168.0.2 is-at 8e:17:54:31:99:e7 (oui Unknown), length 28
        0x0000:  0001 0800 0604 0002 8e17 5431 99e7 c0a8
        0x0010:  0002 feb5 2946 9994 c0a8 0001
18:31:29.326279 IP (tos 0x0, ttl 63, id 11901, offset 0, flags [DF],
  proto ICMP (1), length 84)
    10.0.0.2 > 192.168.0.2: ICMP echo request, id 10815, seq 1, length 64
        0x0000:  4500 0054 2e7d 4000 3f01 4280 0a00 0002
        0x0010:  c0a8 0002 0800 d6cb 2a3f 0001 71cc 8f5e
mininet@mininet-vm:~/project5$
```

Please start early to allow plenty of time for questions on Piazza should you run into difficulties.

What you must submit

For your project submission on Canvas, please turn in two files: `commands.txt` and your project report `report.pdf`. Please do not change or upload `AS352.py`. The questions for the report are listed below. **Please compress the files into a single Zip archive before uploading to Canvas.** Only one team member must submit.

Project report. Please answer the following questions for your project report.

1. Team details: Clearly state the names and netids of your team members (there are 2 of you).

2. Collaboration: Who did you collaborate with on this project? What resources and references did you consult? Please also specify on what aspect of the project you collaborated or consulted.
3. Briefly discuss how you implemented each function: setting up interface configurations, setting up default routes, and setting up the routes for each destination.
4. Is there any portion of your project that does not work as required in the description above? Please explain.
5. Did you encounter any difficulties? If so, explain.
6. Describe **one** technical observation or facts you learned while working on this project. Please answer in specific and precise terms. Your observations could relate to topics involving the network layer in general, your implementation of network configuration in this project, the specific commands you used, or other topics that are relevant to your implementation of this project. **Please ensure your responses are clear, specific, and technical.**

Contact the course staff on Piazza if you have any questions.