TITLE
Project 3: Can you set up your own IP network?

DUE
Mon 04 May 2020 17:00

NUMBER OF RESUBMISSIONS ALLOWED
Unlimited

ACCEPT RESUBMISSION UNTIL
Mon 04 May 2020 17:00

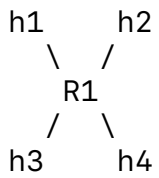
Project 3: Can you set up your own IP network?

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`, `h2`, `h3`, and `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 correct IP addresses for all of the interfaces, and the correct routing table entries for all of 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 ref. [1]) to allow you to create a small network of hosts and routers than can run inside of your machine. You can read more about mininet from ref. [2].

First, download the standard mininet virtual machine (VM) from here:
mininet 2.2.2., ubuntu 14.04 (500+ MB)
`https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-i386.zip`

Set it up and play around according to the instructions:
`http://mininet.org/vm-setup-notes/`

Download and install a *virtualization* program (we recommend that you use **VirtualBox**).

By the end of this project, you must be able to log into the VM using `ssh` and transfer files into the VM (if not, follow the instructions described in the link above!)

STEP 2: Play with the topology Python file

Run the attached **network topology file**, `AS352.py`,
which invokes mining's *Python API* to create a network topology
with **4 endpoints (hosts) and 1 router**.

Note the use of `sudo` while running the program.

```
mininet@mininet-vm:~/project3$ 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 lface
*** 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.

A few examples of playing around with the mininet topology:

Running the `ls` command on host `h1`

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

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

```
h1 <command>
```

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

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

Inspecting 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**,
since it is a native mininet command.

You can even run **commands** on the router!

Mininet is cool because you get to treat both endpoints and the router as standard Linux machines on which you get to do “regular” things.

Let’s **dump the routing table** on `r1`.

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

There is nothing in the routing table on `r1`.
As you may have guessed, that is one of the things that you will fix in this project.

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 mininet command, `source`,
to run all the commands in sequence, one after another.

```
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 at
http://mininet.org/walkthrough/#interact-with-hosts-and-switches

STEP 3: Setting up your network

Now we come 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.**

(1) Set up the IP addresses of all interfaces.

You will use the following IP addresses for each interface:

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

(2) Set up the default routes for the hosts

The hosts `h1`, `h2`, `h3`, and `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 toward the router.

For example, all traffic on host `h1` must use the interface/device `h1-eth0` to reach any destination (other than itself).

Use the `ip route` command to do this.

If you need help understanding the command, try running `ip route help`.

You will set up **4 default routes**, **one for each endpoint** (`h1`, `h2`, `h3`, and `h4`).

(3) Set up the routes on the router

The last remaining piece is having the router interconnecting the endpoints match the packets with the correct outgoing network interface.

The forwarding table entries on router `r1` conceptually must look like:

```
Destination IP address -> Port
h1 (10.0.0.2) -> r1-eth1
h2 (192.168.0.2) -> r1-eth2
h3 (10.0.0.4) -> r1-eth3
h4 (192.168.0.4) -> 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**.

(4) Test your network

You've done all the hard work.

Now test whether 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 route to any valid IP address inside of the network.

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

```
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=168 ms
64 bytes from 192.168.0.4: icmp_seq=2 ttl=63 time=0.046 ms
64 bytes from 192.168.0.4: icmp_seq=3 ttl=63 time=0.044 ms
^C
--- 192.168.0.4 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 192.168.0.4
traceroute to 192.168.0.4 (192.168.0.4), 30 hops max, 60 byte packets
 1 10.0.0.1 (10.0.0.1) 0.018 ms 0.003 ms 0.003 ms
 2 192.168.0.4 (192.168.0.4) 0.008 ms 0.003 ms 0.004 ms
mininet>
```

Note that if you received the error

`traceroute: command not found mininet`"

— please install `traceroute` using

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

Note that there are **2 hops** — the **router** and the **destination** of the `traceroute`,
with the **IP address** of the **router interface** connected to `h1` showing up on **row 1**.

PROJECT TESTING

As part of your submission, you will turn in two text files:

`commands.txt`
`README`

Once you have set up all of the interfaces and routes,
put all of your commands into a file named `commands.txt`.
Please do not modify `AS352.py`.

When your commands file is run by typing `source commands.txt`, it must produce outputs that show successful connectivity for any subsequent `ping` or `traceroute` between two valid IP addresses in the network.

We may test connectivity between any or all pairs of endpoints and the router.
We will only use `ping` and `traceroute` to test connectivity.

```
*** 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>
```

We will run only your `commands.txt` with an unmodified `AS352.py`. We will not accept any modifications to `AS352.py`. **Mininet** should not crash when running your `commands.txt`.

README FILE

In addition to your programs,
you must also submit a `README` file with clearly delineated sections for the following:

0. Please write down the full names and NetIDs of all your team members.
1. Briefly discuss how you implemented each functionality: setting up interfaces, setting up default routes, and setting up per-destination routes.
2. Are there known issues or functions that aren't working currently in your attached code? If so, explain.
3. What problems did you face developing code for this project?
4. What did you learn by working on this project?

SUBMISSION

Turn in your project on Sakai assignments. Only one team member needs to submit. Please upload a single ZIP file consisting of `commands.txt` and `README`.

IMPORTANT NOTES

Mininet has a 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.

Since the aforementioned mininet VM does not comes with a pre-installed graphical interface, using this VM will require you to get comfortable with the Linux terminal/command line.

You are free to install mininet separately on a machine with a graphical interface (note that it requires `sudo` privileges to be installed and run); however, your commands must work with the VM image listed above.

The lack of a graphical interface also means that you may ned to use multiple terminals to run things inside the VM. Terminal multiplexers like `tmux` and `screen` will be very helpful here.

You may be surprised 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.

(That is not the case with this network, of course.)

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.

We recommend using `tcpdump` to sniff the `r1-eth2` interface on `r1`, and dump the sniffed packets to a file named `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 the output of the `tcpdump`:

```
mininet@mininet-vm:~/project3$ 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:~/project3$
```

START EARLY to allow plenty of time for questions on Piazza should you run into difficulties.

References

[1] Linux namespaces. https://en.wikipedia.org/wiki/Linux_namespaces

[2] A network in a laptop: Rapid prototyping for software-defined networks. http://conferences.sigcomm.org/hotnets/2010/papers/a19-lantz.pdf