# Programming Assignment: MPTCP

In this assignment, you will learn how to create a virtual network on your local machine and test the performance of multipath-TCP.

Mininet allows you to set up a virtual network on your machine, running real kernel, switch, and application code. You can create your own topology and experiment with new networking protocols. The majority of your time in this assignment will probably be spent on setup and configuring MPTCP to work properly. Once the initial setup is complete, answering the report questions should be relatively straightforward.

*Collaboration policy:* Each student should submit an individual assignment and complete all steps individually. You are free to discuss with your classmates but you may NOT copy code. If you do discuss with any classmates, please include their names in your writeup.

If you have problems, ask through Piazza.

## Mininet Setup

1. Download the required components.
    a. Download Virtualbox: https://www.virtualbox.org/wiki/Downloads

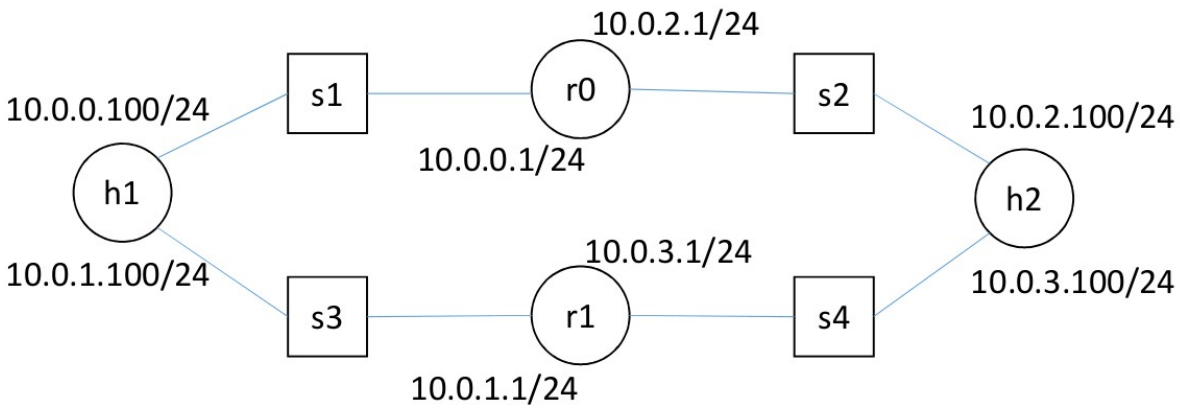
    b. Download the provided virtual machine (VM): http://www.cs.ucr.edu/~jiasi/teaching/cs204_spring20/ubuntu-disk001.vmdk (2.5 GB file, sha1sum b63271f0cddde07c941729b91e6e1fe5d44d0189). This VM contains Ubuntu 18.04.4 LTS server edition, with Mininet, MPTCP, Wireshark, and other minor utilities/files. Create a new machine for Ubuntu 64-bit, and select "Use an existing virtual hard disk file" when prompted, to load the downloaded .vmdk.


    The username is `mininet` and the password is `mininet`.
    c. Configure the Virtualbox so that you can access it through SSH: https://github.com/mininet/openflow-tutorial/wiki/Set-up-Virtual-Machine
    It may already be configured for you. From this point on, you can minimize Virtualbox and interact with it purely through SSH from the host. Don't forget the `-X` option for graphical forwarding.
    *Note*: If you are using a mac, you may need to download Xquartz for x11 forwarding to work correctly.
    *Note 2*: If you have having problems SSH-ing to the host (e.g., connection denied), check that the host and the VM are on the same subnet.

2. Learn how to run a basic topology.
    a. Set up a sample topology:
    https://github.com/mininet/mininet/blob/master/examples/linuxrouter.py
    Run it with `python linuxrouter.py`. This version has 3 hosts, 3 switches, and 1 router, as described in the file comments. Check that the h1, h2, h3 hosts can ping each other (use the `xterm` command in the Resources section to open up a new window for each host).

    b. You can also run Wireshark before you start Mininet. Once you start Mininet, the newly created interfaces will show up in Wireshark, so you can inspect their traffic. This will probably be helpful throughout the assignment.

3. Customize the topology
    a. Create the topology shown below with the IP addresses as indicated. Starting from the `linuxrouter.py` example may be helpful. Note that the `r0` and `r1` routers are actually hosts from Mininet's point of view, which you should set up as `LinuxRouter` objects (following the example in `linuxrouter.py`). `s1`, `s2`, `s3`, and `s4` are switches. For the remainder of this assignment, when we refer to hosts, we mean only only `h1` and `h2`.




    b. Limit the outgoing bandwidth of each interface of each host to 50 Mbps (see the `tc` command in Resources section).

**MPTCP Setup**

1. Enable MPTCP
    a. Setup the appropriate variables using the sysctl command (see http://multipath-tcp.org/pmwiki.php/Users/ConfigureMPTCP). The important variables are net.mptcp.mptcp_enabled (set as 1) and net.mptcp.mptcp_path_manager (set as fullmesh).

2. Configure the MPTCP routing tables
    a. Follow the instructions at http://multipath-tcp.org/pmwiki.php/Users/ConfigureRouting. The automatic configuration script provided is also good but sometimes may not work. You can write your own script using the commands given.

3. Spend some time verifying your topology and MPTCP setup (this is probably the most time-consuming step of this assignment). Use Wireshark to inspect the traffic on the switches and verify that traffic is flowing on the appropriate links. For example:
    a. Can hosts ping each other? Can each host ping its gateways?
    b. When conducting a file transfer between the hosts, is MPTCP working correctly, with traffic flowing on both path 1 and path 2?
        - See how to set up simple Python web server in the Resources section below, and use Wireshark to inspect the traffic.

**Resources**

This is only a barebones guide to the setup. You may find the following resources and commands useful: https://github.com/mininet/openflow-tutorial/wiki

**Handy commands**

- Check all available interfaces: `ifconfig -a`
- Enable interface eth0: `ifconfig eth0 up`
- Check current routing: `route -n` or `ip route show`
- Analyze network traffic: `wireshark`
- Run a simple web server: `python -m SimpleHTTPServer 80`
- Limit outgoing rate on eth0 interface: `tc qdisc add dev eth0 root tbf rate 1mbit burst 1mbit latency 1ms`
- Time webpage download time `time wget -pq --no-cache --delete-after www.cnn.com`

**Mininet-specific commands** (run from Mininet)

- Ping between hosts h1 and h2: `h1 ping h2`
- Any command you want to send to host h1: `h1 <command>`
- Open up a new terminal for host h1: `xterm h1`

**Mininet topology configuration** (e.g., in linuxrouter.py)

- Run a command on h2: `net['h2'].cmd('ip rule add from 10.0.2.100 table 1')`
- Set the IP address of a specific interface on h2:

```
net['h2'].setIP('10.0.3.100/24',intf='h2-eth1')
```

**Report Questions**

**Q1. Does MPTCP improve file transfer performance?**

Set up the web server on `h2` and have `h1` act as the client and try download the Ubuntu ISO (in the `~/www` folder) from `h2` (use the default 10.0.2.100 address). Try this download in two ways: (a) **MPTCP enabled** (two subflows, traffic on path 1 and one on path 2), and (b) **MPTCP disabled** (no additional subflows created, traffic on path 1 only).

Record the trace using Wireshark. To plot the results, use Statistics > I/O Graph and set the filters appropriately to display the relevant traffic in bits/second (i.e., with Wireshark filter `ip.src==XXX` and `ip.dst==YYY`). Create two plots, corresponding to scenario (a) and (b) above. Each plot should have one line for traffic on path 1, and one line for traffic on path 2.

*To include in report:* The two plots above. Also answer: does MPTCP increase the total throughput or decrease the total download time? Does this make sense, why or why not?

**Q2. Does MPTCP improve (approximated) web browsing performance?**

Set up the web server on `h2`, and have `h1` act as the client and try download a webpage (`index.html` in the `~/www/BBC_190420` folder) from `h2` (use the default 10.0.2.100 address). Use the `wget` command in the Resources section. Try this download in two ways: (a) **MPTCP enabled** (two subflows, traffic on path 1 and one on path 2), and (b) **MPTCP disabled** (no additional subflows created, traffic on path 1 only). Record the real time output from `time wget`. Also simultaneously run Wireshark and count the number of objects requested over each interface using Wireshark (hint: use Wireshark filter `http.request.method=="GET"`). Create a table of these results as shown below.

| | Wget time | Fraction of objects on path 1 |
|---|---|---|
| MPTCP enabled | | |
| MPTCP disabled | | |

*To include in report:* The result table as shown above. Also answer: does MPTCP help reduce the page load time? What fraction of objects are requested over each path? Does this make sense, why or why not? Is `wget` a reasonable approximation of a web browser, or would you expect a real web browser to have different performance?

**Submission Instructions**

Submit the answers to the above "Report Questions" section in a written report. Include any Python code you wrote in an Appendix.

**Due date**: Sunday May 3, 11:59pm, through iLearn