Advanced Computer Networks (2021)

Lab0: Getting Familiar with Mininet

Type: Individual
Maximum points: 0
Submission: None
Deadline: Monday November 8, 2021
Contact: vu.acn.ta@gmail.com

The goal of this lab is to learn the basics of Mininet. Later assignments will be largely based on these basics so it is very important that you complete it.

1. Set up the course virtual machine

The first step is to set up the virtual machine (VM) you will be using for the rest of this course. This will make it easy to install all dependencies for the labs, saving you the tedium of installing individual packages and ensuring your development environment is correct.

Install Vagrant

Vagrant is a tool for automatically configuring a VM using instructions given in a single "Vagrantfile". For macOS or Windows users, please use the correct download link here: https://www.vagrantup.com/downloads.html. If you are a Windows user you may be asked to restart your computer at the end of the installation. Click "Yes" to do so right away, or restart manually later, but do not forget to do so; otherwise, Vagrant will not work. For Linux users, first, make sure your package installer is up to date by running the command `sudo apt update`. To install Vagrant, you must have the "Universe" repository on your computer; run `sudo apt-add-repository universe` to add it. Finally, run `sudo apt install vagrant` to install vagrant.

Install VirtualBox

VirtualBox is a VM provider (hypervisor). For macOS or Windows users, please use the correct download link here: https://www.virtualbox.org/wiki/Downloads. The links are under the heading "VirtualBox 6.x.x platform packages". For windows users, use all the default installation settings, but you can uncheck the "Start Oracle VirtualBox 6.x.x after installation" checkbox. For Linux users, you can install VirtualBox by running the command `sudo apt install virtualbox`. This will also install the VirtualBox application with GUI on your computer, but you should never need to run it.

Install Git (and SSH-compatible terminal on Windows)

Git is a distributed version control system. For macOS or Windows users, please use the correct download link at: https://git-scm.com/downloads.

For macOS: Once you have opened the `.dmg` installation file, you will see a Finder window including a `.pkg` file, which is the installer. Opening this normally may give you a prompt saying it cannot be opened because it is from an unidentified developer. To override this protection, instead right-click on the `.pkg` file and select "Open". This will show a prompt asking you if you are sure you want to open it. Select "Yes". This will take you to the (straightforward) installation.

For Windows: You will be given many options to choose from during the installation; using all the defaults will be sufficient for this course (you can uncheck "View release notes" at the end). The installation includes an SSH-capable Bash terminal usually located at `C:\Program Files\Git\bin\bash.exe`. You should use this as your terminal in this class, unless you prefer another SSH-capable terminal (the command prompt will not work). Feel free to create a shortcut to it; copying and pasting the executable somewhere else will not work, however.

Linux users can simply install Git using the following command: `sudo apt install git`.

Install X server

You will need an X server to input commands to the virtual machine. For macOS users, please install XQuartz following the instructions at: https://www.xquartz.org. You will need to log out and log back in to complete the installation (as mentioned by the prompt at the end). For Windows users, please install Xming following the instructions at: https://sourceforge.net/projects/xming/. You can use default options and uncheck "Launch Xming" at the end. For Linux users, the X server is pre-installed already (cool!).

Download the course lab package

Please download the course lab package on Canvas: `Files/project/acn21-project-base.zip`
and unpack it to the directory where you want to keep files for this course on your computer. Open your terminal and navigate to the directory, you will be able to see six placeholder folders with name `lab0` through `lab4`. Together with these folders, you also see a file named `Vagrantfile` and a folder named `scripts` which will be used to set up the virtual machine. Please note that you should not change any of the structure of the course directory; otherwise your code may not compile correctly later. Every time when a new lab assignment (except the first `lab0`) which is already available in the code base becomes available, you will receive a compressed folder with name `lab*.zip`. You should decompress it and use it to override the placeholder folder under the `acn21-project-base` directory.

Provision virtual machine using Vagrant

Before you provision the virtual machine, it is required that you enable Vagrant to resize the VM disk space by running `vagrant plugin install vagrant-disksize`. From the directory with the `Vagrantfile` file, run the command `vagrant up` to start the VM and provision it according to `Vagrantfile`. You will likely have to wait for a while (e.g., 30 mins). You may see warnings/errors in red, such as `default:stdin: is not a tty`, but you should not have to worry about them. Once your VM is up successfully, the following commands will allow you to stop the VM at any point:

- `vagrant suspend` will save the state of the VM and stop it.
- `vagrant halt` will gracefully shutdown the OS and power down the VM.
- `vagrant destroy` will remove all traces of the VM from your system. If you have important files saved on the VM (e.g., your lab solutions), DO NOT use this command.

Additionally, the command `vagrant status` will allow you to check the status of your VM in case you are unsure (e.g., running, powered off, saved,...). You must be in some subdirectory of the directory containing `Vagrantfile` to use any of the commands above, otherwise Vagrant will not know which VM you are referring to.

Test SSH to the VM

Running `vagrant ssh` from your terminal allows you to connect to your VM. This is the command you will use every time you want to access the VM. If it works, your terminal prompt will change to `vagrant@vagrant:/vagrant`. All further commands will execute on the VM. The course directory (`/vagrant`) is actually shared between your host OS and the VM. This shared folder is especially useful. You do not need to copy files to and from the VM. Any file or directory in the directory where `Vagrantfile` is located is automatically shared between your computer and the VM. This means you can use your IDE of choice from outside the VM to write your code and build/run within the VM. The command `logout` will stop the SSH connection at any point.

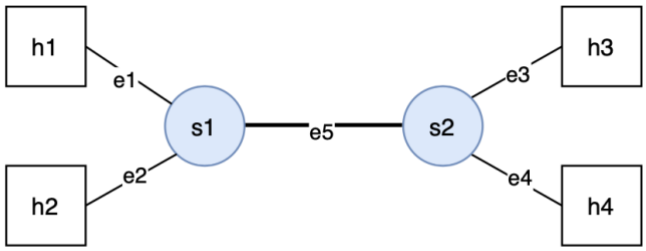
2. Basic Mininet usage

Mininet is pre-installed in the VM you have just set up. Please complete the online Mininet walkthrough at http://mininet.org/walkthrough/. In some parts of the walkthrough, you may see software-defined networks. We will discuss it later in this course, so you do not need to understand what they mean right now. You just need to know how to run and interact with Mininet.

Note: In the second part of the walkthrough, you are asked to build a customized topology in Mininet. The programming language used is Python. If you have no experience with Python, please refer to the Python tutorial at https://docs.python.org/2.7/tutorial/index.html to get yourself familiar with the basics of Python. (We use Python 2 for compatibility reasons.)

3. Build a customized network topology

Using the Mininet APIs you have just learnt, build the following topology. Hosts `h1`-`h4` are represented by squares and switches `s1` and `s2` are represented by circles. The name of the devices you build in Mininet should match the names in the diagram. The hosts are assigned IP addresses `10.0.0.x` where `x` should match the host number (1-4). The properties (bandwidth, delay) of the links are specified as: (15Mbps, 10ms) for links `e1` - `e4` and (20Mbps, 45ms) for link `e5`.



e1-e4: 15Mbps,10ms e5: 20Mbps,45ms

One you have completed the topology file, launch it in Mininet using the following command:

```
sudo mn --custom /path/to/topo/file --topo=topo name --link=tc
```

4. Measure the performance of the network

Use xterm to start terminals for the hosts in the network.

```
mininet> xterm h1 h3
```

Now, please measure the latency and throughput between hosts `h1` and `h3` using `ping` and `iperf`, respectively. For `ping` you should send 20 or more packets and average the results and for `iperf`, you should measure for 20s or longer. Measure with both TCP and UDP connections.

```
h1$ iperf -s (-u)
h3$ iperf -c ip.to.server -t 20 (-u)
```

5. Effect of multiplexing

In the above measurement we only have one connection going at the same time. Now, let us try with simultaneous connections where we let `h1` talk to `h3` and `h2` talk to `h4` at the same time. Try to predict the latency and throughput. Use `ping` and `iperf` to measure the latency and throughput of the two connections. Check if your predictions are correct and think about why or why not. What if we have two connections all destined to `h4` (i.e., `h1`-`h4` and `h2`-`h4`)?