The CORE emulator

Dino Lopez Pacheco
dino.lopez@univ-cotedazur.fr

1 Introduction

In this course, we'll be using CORE - *Common Open Research Emulator* (https://coreemu.github.io/core/), which runs on Linux systems. In this lab, we will explore this tool and learn how it works.

2 Installing & Configuring a Linux VM

Since the installation of CORE is a process that can be complicated, we have installed CORE in a Linux VM that you will use for your courses at Polytech, and made available to you by the department.

Even if you have a Linux machine (e.g. dual boot), you need to install the VM provided by the department.

If you have a Mac computer with an M1 chip, you must follow the VM installation procedure in section 2.2.

For all other cases (Intel chip-enabled PCs and Macs), install the Linux VM using the steps in section 2.1

2.1 Linux VM installation (only PC and Mac with Intel chips)

First, make sure you have at least 35GB free disk and 8GB RAM. Then, install the latest stable version of the ***VirtualBox*** software and import the Linux Virtual Machine, following the procedure below.

Installing VirtualBox

To install VirtualBox, all you have to go to the official website: https://www.virtualbox.org/
The installation is carried out in two steps:

- Installation of the "VirtualBox platform packages". Download and install the appropriate software for your operating system.
- Installation of the "VirtualBox Oracle VM VirtualBox Extension Pack" which allows you to add functionality to the software (and which is not dependent on the system on your machine). Once the Extension Pack is downloaded, double click it, the VirtualBox software will automatically open and guide you through the installation process.

Now, all you need to do is recover the image containing the Linux VM.

Retrieving the VM image

Download the VM image by following the link http://images-vm.polytech.unice.fr/PNS_Ubuntu64.ova Note that the size of the VM is about 4 GB, which can be longer or shorter depending on your type of Internet access.

Once the download is complete, double-click on the PNS_Ubuntu64.ova file. A window opens and allows you to consult the configuration of the virtual machine and above all, to install it by pressing the "Import" button (or "Import", depending on the language of your system).

VM user account

A generic user account has been created for you to log in. The identifiers are:

- Username: user
- Password: pns

1. Validate the installation of the VM with your lab supervisor

2.2 Installing & Configuring a Linux VM (Mac computers with M1 or M2 chip only)

We offer you an experimental solution since you cannot install the department's VM:

Install UTM (see https://mac.getutm.app/), then download the Ubuntu ARM64 VM available here http://i3s.unice.fr/~castillon/share/UbuntuCORE.utm.tar.gz Double-click the downloaded file to unzip the VM (you will get a .utm file) and, finally, double-click the unzipped file to import the VM.

The login/mdp of the ARM64 VM is "student" / "student" (without the "").

ARM64 VM Network Activation - UTM

Once the VM is installed, run and log into the VM. Open a terminal and type the command "$ sudo su" to become root.

Use the command "# ip a s" to find the name of your network interface that gives you access to the Internet. Then, get an IP address for that interface with the command "# dhclient -v enp0s1" (assuming that you found the name "enp0s1" with the first command). Note that you must do this every time the VM (re)boots.

For a permanent configuration of the network, see with your Lab supervisor.

Installing Packages

As root, run the commands "#apt update & apt install isc-dhcp-server iperf3"

2.3 Known Bug (Only PC and Mac with Intel Chip)

- Mac computers
    - You encounter an error when running the VM, with a message "Kernel driver not installed" (see image at this URL - https://www.howtogeek.com/wp-content/uploads/2020/02/Kernel-Driver-Not-Installed-rc-1908.png). Follow this procedure to fix the https://www.howtogeek.com/658047/how-to-fix-virtualboxs-"kernel-driver-not-installed-rc-1908-error/
- If you are unable to fix your VM issues:
    - see the systems team of the Templiers campus (Mr. Christophe COROYER (O+224) - Christophe.COROYER@univ-cotedazur.fr or Mr. Thierry NEDELEC (O+212) - nedelec@polytech.unice.fr)

3 Adapt the VM to your machine (only PC and Mac with Intel chip)

3.1 Installing Guest Additions

To get a properly sized screen (among other services), you'll need the *Guest Additions* modules in your VM.

The VM we provided has already a version of Guest Additions. However, if the VM has been configured in a different environment than the one you have in your laptop, it is possible that Guest Additions is disabled and its reinstallation may be necessary.

Turn on your VM. Then open the terminal and run the command

```
$ lsmod | grep vbox
```

If you see the following 2 lines (the numbers may differ), the Guest Additions are working correctly and you can request the validation of the Guest Additions installation with your lab supervisor.

```
vboxvideo 36864   0
vboxguest 364544        5
```

If not, you need to reinstall the Guest Additions. To do this, follow steps 3 to 5 in this tutorial https://www.tecmint.com/install-virtualbox-guest-additions-in-ubuntu/

At the end of the installation, restart your VM for the Guest Additions modules to load.

2. Validate the installation of Guest Additions with your lab supervisor

3.2 Creating a sharing space between your VM and your laptop

Often, it is very useful to have the files and folders of your laptop accessible from the VM (for example, you can write a script on your favorite Windows editor and run it on the VM).

You now need to set up a shared folder between your laptop and your VM (e.g. a folder on your Desktop on Windows). To do this, follow the tutorial available at this address https://youtu.be/9-teQnZ8LEY . Once you have configured the shared folder, you need to access it through the file explorer (folder icon on the left bar, /media/sf_Your_Folder_Name folder) because in our Ubuntu version, shared systems are not displayed directly on the desktop.

3. Validate the creation of the shared folder with your lab supervisor

4 Running CORE

4.1 "1 LAN" topology

Now that your VM is ready, let's start exploring CORE. Launch a terminal on the VM, then become "root" with the command:

```
$ sudo su
```

On the root prompt, run the command

```
# core-daemon
```

This will display a few lines without returning the prompt. It is the *backend* of CORE that runs and will be the real responsible for creating the virtual networks. Thus, you need to leave this window open and open a new tab on the terminal to retrieve a standard user prompt.

Open a new tab (click on the "tab" icon at the top left of the terminal window or press Ctrl+Shift+T). Now run the CORE client (the *frontend)* with the command

```
$ core-gui
```

Or run "$ core-gui-legacy" if you're on the UTM VM

Check that everything is going well. The CORE GUI should be displayed and on the last terminal, you should have the line "Connecting to "core-daemon" (127.0.0.1:4038)..." connected. ».

To create a virtual network, you can add level 3 devices (network-layer virtual nodes button, "router" icon on the left bar), level 2 devices (link-layer nodes button, switch icon), and links

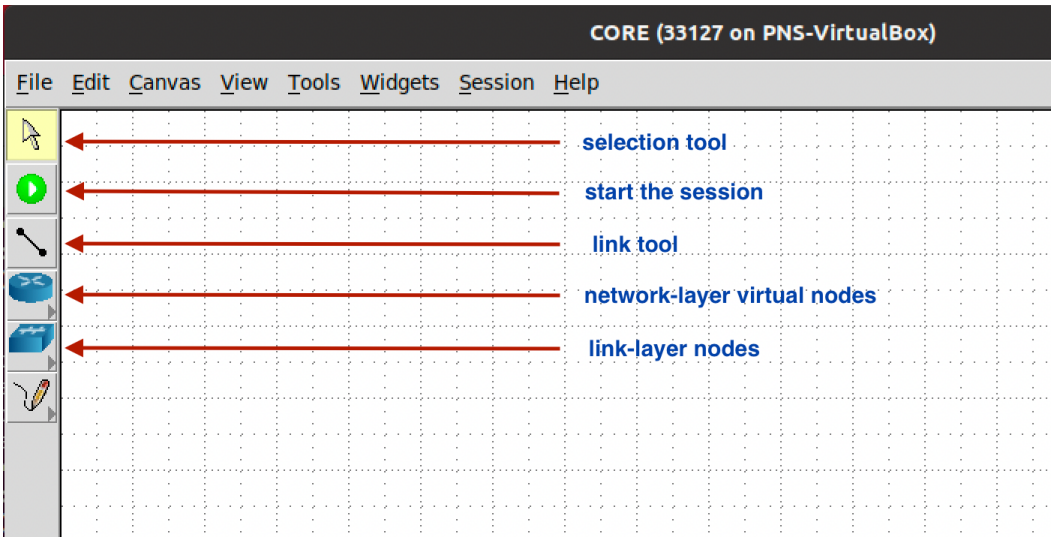

Figure 1. CORE GUI

between devices (link tool button). Here is a screenshot to help you spot these elements.

Now, create the network topology below. Click on "network-layer", then click on "host" ("server" icon). Click on the workspace to place 2 nodes in it. Then, click on link-layer", then "ethernet switch" (2nd icon from left to right). Click on the workspace to place a switch. Finally, after clicking on "link tool", create a link between the nodes as shown in the figure (click on the "host" and by holding down the mouse, join the "switch").

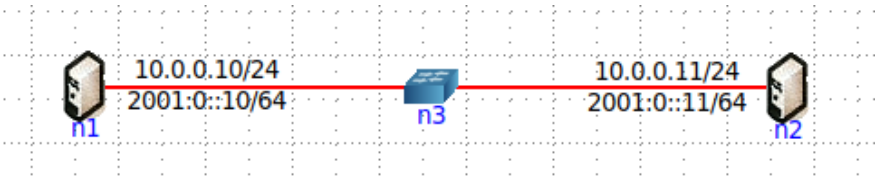

Figure 2. Network topology with 2 servers and 1 switch

Once the network topology is designed, you can deploy it by clicking on the "star the session" button. Red frames will be drawn on each network device while they are deploying, and then the frames will turn green if everything goes well (and meaning you can now interact with the virtual network).

To interact with the network, we can double-click on level 3 nodes (i.e. devices created with the "network-layer virtual nodes" button) to obtain a shell (terminal).

4. In Linux, the configuration of a network interface is achieved primarily with the “`ip`” command. Double-click on the servers (n1 and n2 in Figure 2) and with the command "`ip addr show`» show the configuration of the "eth0" interface. Does the IPv4 address shown by CORE (10.0.0.10 for n1 in the Figure 2) is the same as you get with the "ip" command?

5. Let's now do a "ping" to validate the accessibility of "n2" from "n1". Open a terminal for "n1" and run the command "ping -c5 10.0.0.11". Also ping to the address 10.0.0.100 (non-existent address). Give the results of the command and explain the output of the ping when the targeted machine exists.
   5.1. A ping is a very simple application where a host sends a request (technically, an *ICMP echo request* message) and the receiver sends a response (*ICMP echo reply message*)
6. Open a terminal on "n2" and clear its current IPv4 address with the command "ip addr del 10.0.0.11/24 dev eth0". Check with the command "ip a s" (short for "ip addr show") that the IPv4 address is gone. Then give it the address 10.0.0.100 with the command "ip addr add 10.0.0.100/24 dev eth0". Finally, check by command line that the new address is present. Read the IPv4 configuration given of "n2" shown by CORE. Do you see the importance of using the command line to verify your test network configuration?
7. Prove with a ping from "n1" that the new address of "n2" is operational
8. Stop the virtual network by clicking the "stop the session" button.

4.2 "Multi-LANs" topology

9. After you stop the virtual network from running, change your topology to add a router and a 3rd server, as shown in the figure above.

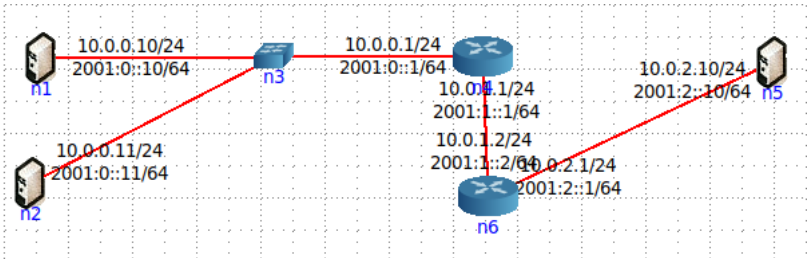

Figure 3. Multi-LAN topology

10. Start the network deployment, wait a few seconds after the network is deployed, and verify that you can ping "n5" from "n2", in the figure Figure 3. Show with screenshots that the ping works as expected.
11. Stop the virtual network, then exit the CORE GUI, but let the daemon running.

After doing the last exercise, you can see the "magic" of CORE, which provides full connectivity between elements without you having to do anything.

5 Explore some network services

12. Download the file "topo-2lans.imn" given as a resource for this lab.
    12.1. Keep this file in a folder with no accents or spaces in the path (e.g. keep your file in $HOME/Documents/)
    12.2. Launch the CORE GUI again and through the "File -> Open..." and open the downloaded .imn file.
    12.3. Start creating the virtual network.

5.1 Obtaining an address

13. Open a terminal on "n2" and run the command "# ps aux". Check that a line starting with "root" and ending with "dhcpd" exists, continue with the next exercise. If not, close everything, exit CORE completely, and restart your VM before trying this exercise again.
14. Open a terminal on "n1" and give its IPv4 address if it exists. Give the command and output you got.

15. On the same terminal of "n1", run the command "# dhclient -v eth0". Next, check the IPv4 address. Give the obtained information.

5.2 Firefox browser test

16. Open a terminal on "n3" and then run the command "# python3 -m http.server 80" to create a minimal HTTP server. The command does not return the prompt. Let your order run.
17. Close any open Firefox windows. Go to the terminal of "n1" and run "# firefox". If you see an "Error: cannot open display::0 ", execute the "xhost +" command on a terminal on your VM (i.e. outside the virtual network). Then, try the "firefox" command again on the "n1" terminal.

    a) It is normal for Firefox not to find an Internet connection since our virtual network has no output to the external world.

18. Once Firefox is open from "n1", type the IPv4 address of "n3" on the address bar. Verify that the HTTP server is working properly by browsing with Firefox on the "n3" file system.
19. Open a 2nd terminal on "n3" and run the command "# wireshark". On the window that opens, double-click on "eth0". The Wireshark app will start a capture of all traffic seen by the "eth0" interface of "n3".
20. Continue exploring the "n3" file system from Firefox and try to identify the corresponding network traffic on Wireshark.
21. Close Wireshark, the Firefox browser, and on the terminal where the python HTTP server is running, run Ctrl+c to shut down the server.

5.3 SSH Service Testing

22. Open a terminal on "n3" if it is not already open, then give a password to its root user (who is not the root of the VM) by running the "# passwd" command. You will have to type twice the same password. Even if nothing is displayed on the terminal, what you type will be correctly picked up by the command.
23. Open a terminal on "n2" if it is not already open. Run the command "ssh root@10.0.0.10" to log in to "n3". You will need to accept ("yes") the security certificates and use the root password given on the previous point.
24. Use the "hostname" command to check that you are on "n3". End the SSH session with the "exit" command and return to "n2".

5.4 Bandwidth test with iperf

25. On a terminal of "n3", run an iperf server with the command "iperf -s -i1" (where i1 indicates that we would like to have a synthesis of the statistics 1 time per second). The prompt is not returned. So let iperf run.
26. On terminal "n2", run an iperf client with the command "iperf -c 10.0.0.10". This client will send as much data as possible to a 10.0.0.10 using the TCP protocol. The objective is to see what maximum throughput is reached at the receiver (the iperf server) to determine the capacity of the bottleneck.
27. Wait for the iperf client to finish running and look on the server side at what the throughput (and therefore the bottleneck capacity) is between "n2" and "n3".
28. With CORE, we could also have observed the instantaneous throughput passing through the links and triggered an alarm when this throughput reaches a certain capacity. Click on "Widgets -> Configure Throughput" and declare a threshold of 1000 kbps (i.e. 1Mbps). Then activate the alarm (click on "Widgets -> Throughput").

29. Repeat the experiment with iperf and verify that the alarm goes off correctly when transmitting data. Here you have an easy way to detect the path that network traffic is taking.
30. Close all terminals and stop the virtual network (click on "stop the session"). Stop the CORE daemon by pressing Ctrl+c.