

How to Set Up the Project and Run the CPP Codes

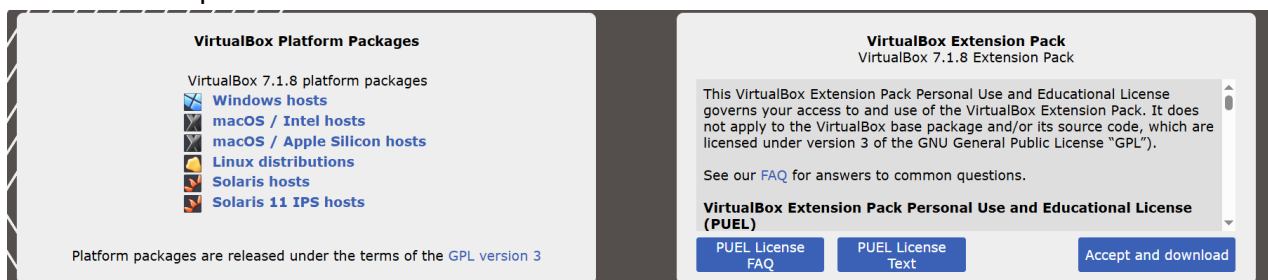
OVERVIEW - Programs and downloads/upgrades you will need:

- Oracle VirtualBox (or any VM program of your choice)
- Ubuntu ISO disk file for the VM
- NS-3 installation in the VM (I used version 3.42)
- Python, Wireshark, and other dependencies
- OpenSSL (built into Ubuntu VM)

PART 1 - Ubuntu VM on Virtual Box:

1. Install Oracle Virtual Box (I installed version 6.1.16):

- a. Go to <https://www.virtualbox.org/> and click on the “download” button under “Get Started” to see the list of OS distributions for VirtualBox. If you have windows, download the windows version, etc. You will also need the extension pack



- b. I don't think it should matter which version you get, but if you want to download the same version 6.1.16, scroll all the way down and find “Previous Releases” on the right. Click on “VirtualBox older builds”, then scroll down to the older version you want and download the correct OS distribution and the extension pack for the version.

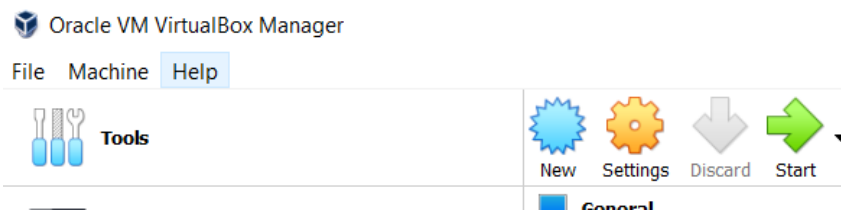


- c. Start the installation for VirtualBox and follow the wizard. After that, go to the extension pack you downloaded and open it up to start installing the extensions. When you get to the “VirtualBox License” section, scroll all the way down for the buttons to be enabled, then click “I Agree.” Use this video to help see the whole installation process: <https://www.youtube.com/watch?v=HC5obz0YPuc>
 - d. With VirtualBox installed, we will now create the Ubuntu VM.
2. Create Ubuntu virtual machine (I used version 20.04):

- a. Go to the Ubuntu website for old releases (<https://old-releases.ubuntu.com/releases/20.04.1/>) and scroll down to download the desktop .iso file for version 20.04.1. This will be used for creating the VM, so keep it in a place where you can find it. The current long-term-support version is 24.02, but I recommend getting 20.04 to ensure the project functionality is duplicated accurately. Note that it will take some time for the iso to finish downloading.

	ubuntu-20.04-preinstalled-server-arm64+raspi.manifest	2020-04-23 15:21	14K
	ubuntu-20.04-preinstalled-server-armhf+raspi.img.xz	2020-04-23 15:20	638M
	ubuntu-20.04-preinstalled-server-armhf+raspi.img.xz.zsync	2020-04-23 17:14	1.2M
	ubuntu-20.04-preinstalled-server-armhf+raspi.manifest	2020-04-23 15:20	14K
	<u>ubuntu-20.04.1-desktop-amd64.iso</u>	2020-07-31 16:52	2.6G
	ubuntu-20.04.1-desktop-amd64.iso.torrent	2020-08-06 15:30	208K
	ubuntu-20.04.1-desktop-amd64.iso.zsync	2020-08-06 15:30	5.2M

- b. Now, open VirtualBox Manager and click “new” (Use this video to also help see the whole VM setup: <https://www.youtube.com/watch?v=IOwlnpWPuj0>)



- c. Name your VM “Ubuntu 20.04” or any name you choose. Make sure the type is Linux and version is Ubuntu (64-bit). Choose the location you want to save it (if you have a D drive with plenty of space, store the VM there). Click “Next”.


← Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

Machine Folder:

Type: 

Version:

Expert Mode **Next** Cancel

- d. Allocate some memory to the VM. The minimum is 4 GB, but I allocated 10 GB. Click “Next”.

← Create Virtual Machine

Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024 MB**.

 MB

4 MB 32768 MB

Next Cancel

- e. The default should be “Create a virtual hard disk now”. Click “Create”.

← Create Virtual Machine



Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **10.00 GB**.

- ☐ Do not add a virtual hard disk
- ☒ Create a virtual hard disk now
- ☐ Use an existing virtual hard disk file

 kali-linux-2024.1-virtualbox-amd64.vdi (Normal, 80.09 GB) 

Create Cancel

- f. Select VDI then “Next”. Then select Dynamically allocated and then “Next”.

Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- ☒ VDI (VirtualBox Disk Image)
☐ VHD (Virtual Hard Disk)
☐ VMDK (Virtual Machine Disk)

Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.


A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

- ☒ Dynamically allocated
☐ Fixed size

- g. Now select the size of the virtual disk. The minimum is 25 GB, but I chose 40 GB. Then click “Create”.

File location and size

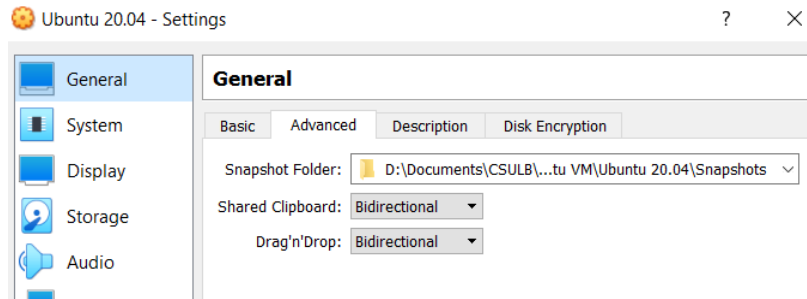
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

C:\Users\krist\VirtualBox VMs\Ubuntu1\Ubuntu1.vdi 

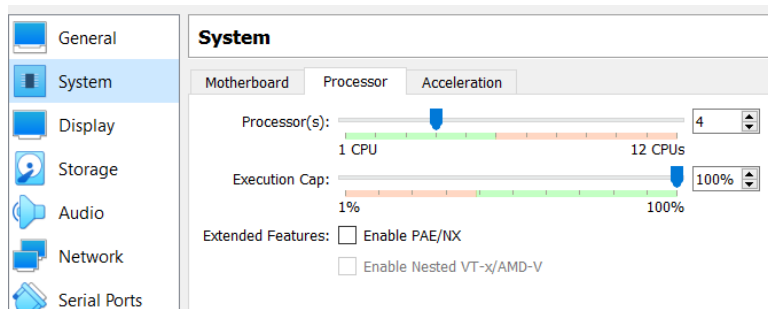
Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.



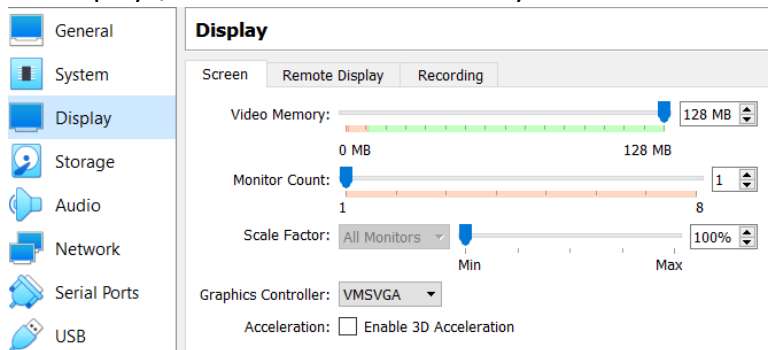
- h. The VM should be created, but before we start it up, we need to do some modifications and also include the Ubuntu ISO.
- i. Select the virtual machine and go to “Settings” (it’s next to the “New” button)
- j. In “General”, go to “Advanced” and select “Bidirectional” for both shared clipboard and drag’n’drop. This will make copy-and-pasting text easier when going back and forth between the VM and your actual computer.



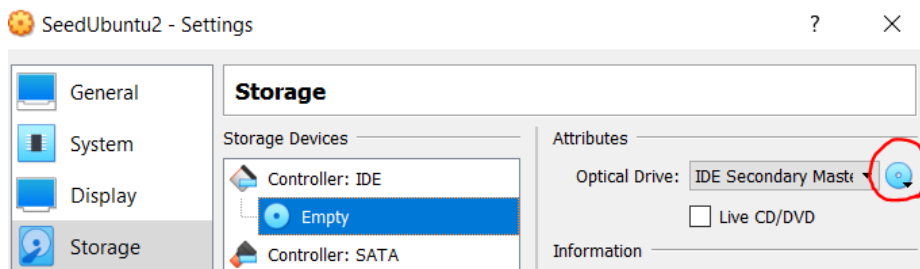
- k. Go to “System” and increase the number of CPUs if you can. 2 is the minimum, and I believe the highest allowed for the Ubuntu VM to properly work is 4, which I set mine to.

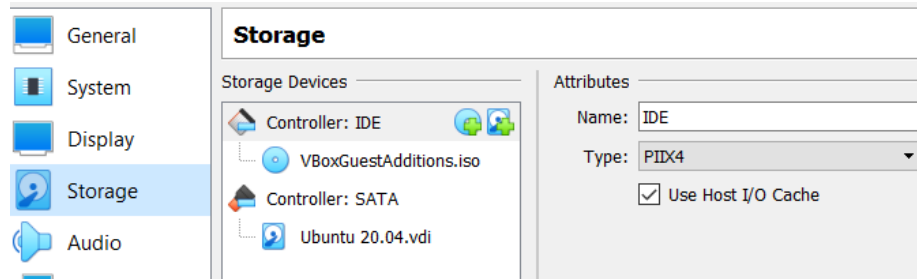


- l. In “Display”, max out the video memory to 128 MB.



- m. Now we just need the ISO. Go to “Storage”, select the empty disk, and click the disk icon next to Optical Drive and select “choose a disk file”. Find where you saved the ISO and select it.

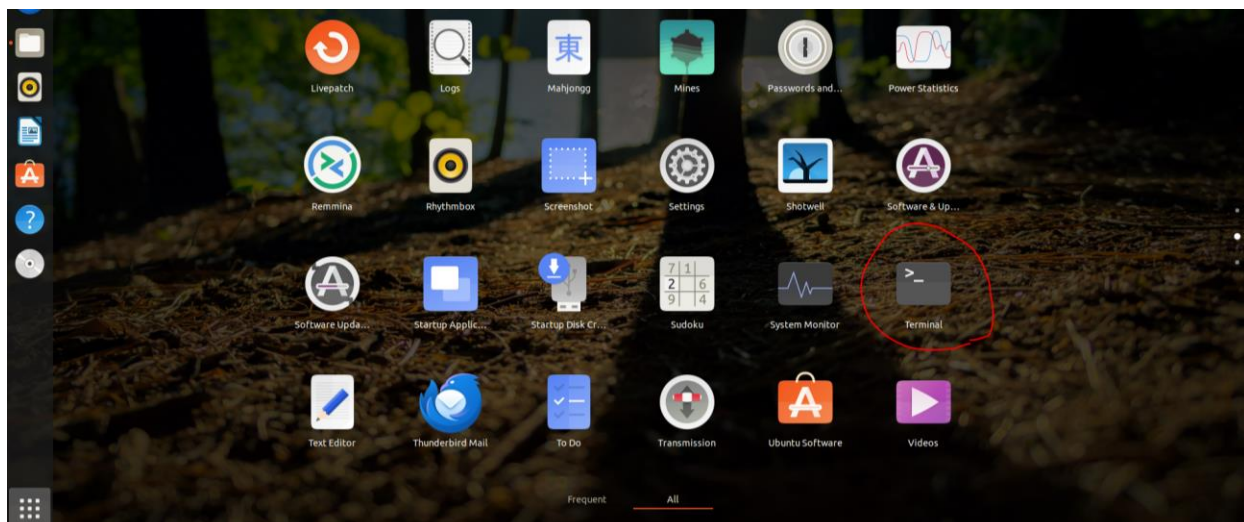




- n. The Ubuntu VM is now set and ready to run. Select your VM and then hit the “Start” button.
- o. The first thing the VM will ask you is to select the start-up disk, so make sure the Ubuntu ISO is selected, then hit start. Follow the Ubuntu installation process (see the video link in part 2b). When you create your Ubuntu account, make sure you remember your username and password.
- p. After installation, it will prompt you to restart the VM, so accept the restart. Once restarted, you can log in to the VM. The VM should now be okay to at least use.

3. Installing Guest Additions:

- a. Now you’ll notice the screen is square-shaped and not taking up the whole VM window when you try to maximize it, so first you need to open the terminal in the VM and do some updates and upgrades. Go to the bottom left corner that looks like a white Rubik’s cube and select “Terminal”.



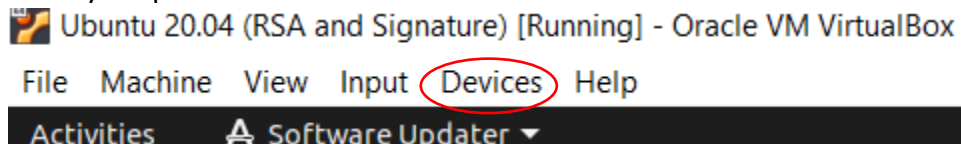
- b. Enter **sudo apt-get update** (you will need to enter your password to continue)

```
kbjackson@kbjackson-VirtualBox:~$ sudo apt-get update
[sudo] password for kbjackson:
```

- c. Wait for it to finish, then enter **sudo apt-get upgrade**. It will ask you “Do you want to continue? [Y/n]”. Enter **y** for yes and let it continue running. This part may take a few minutes to complete.

```
kbjackson@kbjackson-VirtualBox:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 libfprint-2-tod1 libfwupdplugin1 libxmlb1
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
  linux-generic-hwe-20.04 linux-headers-generic-hwe-20.04 linux-image-generic
The following packages will be upgraded:
  binutils binutils-common binutils-x86-64-linux-gnu distro-info-data gir1.2-
  libcryptsetup12 libctf-nobfd0 libctf0 libmagickcore-6.q16-6 libmagickcore-6
  libxml2-dev libyelp0 linux-libc-dev openssh-client poppler-utils ubuntu-adv
34 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 12.1 MB of archives.
After this operation, 115 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

- d. Then enter **sudo apt install build-essential dkms linux-headers-\$(uname -r)**. You may get asked again “Do you want to continue?”. Say yes like before and let it run. Once it’s done, go to Devices and select “Insert Guest Additions CD image”. You will be prompted in the VM to run it, so hit “run”, and then you will have to enter your password to authenticate it.



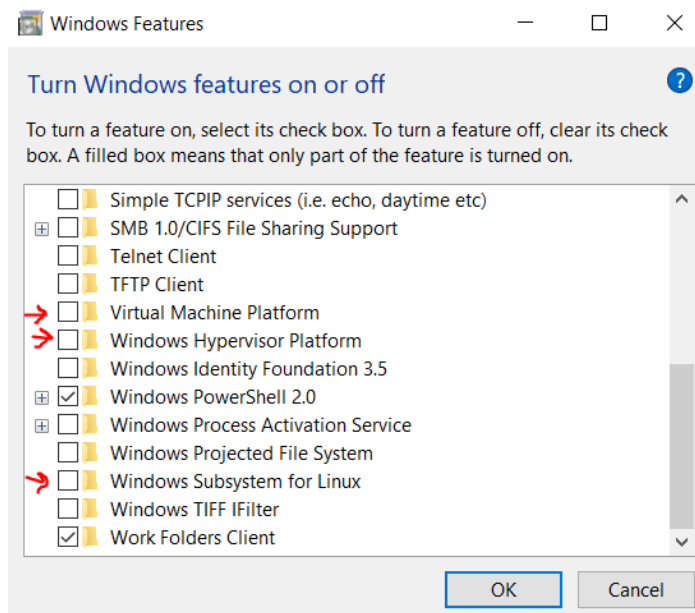
- e. Now the VM needs to be restarted. Go to the upper right corner and select the power button. Drop down the “power off / log out” and select “power off”. It will prompt you to make sure you want to power off, so click the power off button in the prompt. We want to shut down the VM completely to ensure the guest additions will be applied.
- f. Once fully shut down, start the VM again and log in like usual. The VM screen may still look square, so you can restore down the window and then maximize it again, and you should be able to see the changes applied, and now the VM screen is much bigger. Congrats! You have finished setting up the Ubuntu VM! Next step is to install NS-3 and its dependencies needed for the library, especially

Wireshark.

- g. **NOTE:** If you ever try to start the Ubuntu VM again later on and it does not ever start up past the black screen and is stuck, check if the bottom right of your VM window has the green turtle. If so, that means you have Hyper-V on.



- i. Hyper-V conflicts with your Ubuntu VM, so you have open **Turn Windows features on or off** in your windows start menu and then disable **Hyper-V** or **Hypervisor** and **Windows Subsystem for Linux** if that's available, too. If it's there, you might also need to turn off **Virtual Machine Platform** if the VM still does not start up.



- ii. When you start up your VM now, it should be able to load properly. Also, the icon should no longer be the turtle one.



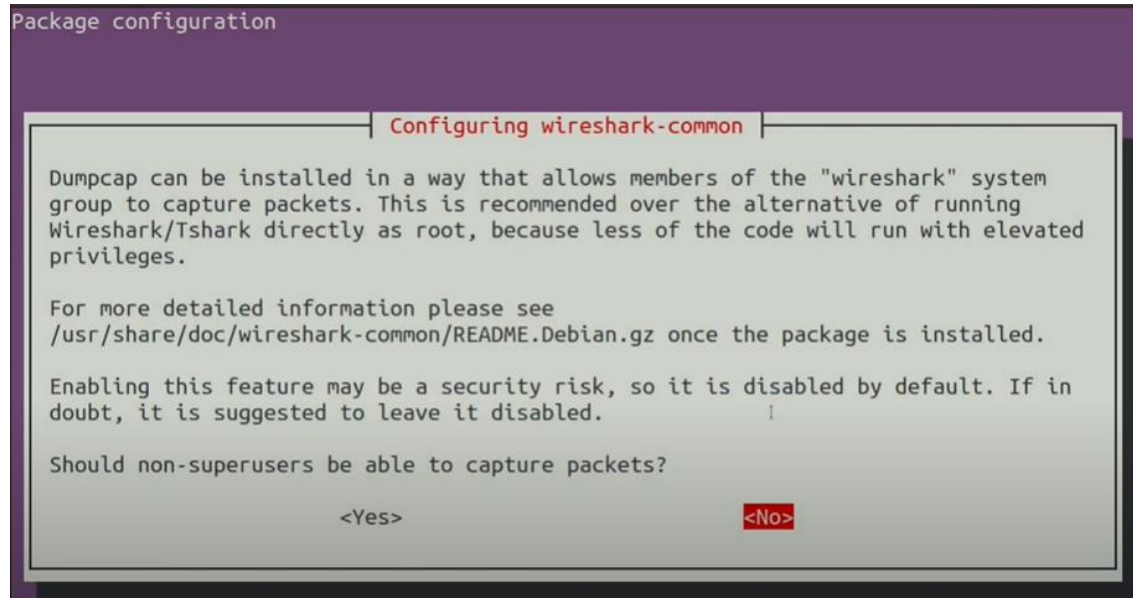
PART 2 - NS-3 and Environment Setup (Wireshark, Python, dependencies):

1. Before installing NS-3, we first need to make sure that we have Wireshark installed since this project will involve using it to see the captured packets in the edge computing models. Similar to installing guest additions for the VM, we will make sure that these dependencies are covered. Please watch this video to also help see the NS-3 installation process: <https://www.youtube.com/watch?v=timruVmsOxl>
2. Open your terminal and enter **sudo apt update** to make sure the VM is up to date.

3. Next, enter this command to install Wireshark and other possible dependencies that may be missing from the previous installation:

```
sudo apt install g++ python3 python3-dev python-dev pkg-config sqlite3 python3-setuptools git qt5-default gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 openmpi-bin openmpi-common openmpi-doc libopenmpi-dev autoconf cvs bzip2 unrar openmpi-bin openmpi-common openmpi-doc libopenmpi-dev tcpdump wireshark libxml2 libxml2-dev
```

4. You will follow a similar prompt "Do you want to continue?". Say yes like usual. Now during this installation, because you are including Wireshark, you will have a special prompt about configuring Wireshark. Use the arrow key to go to <Yes> and hit Enter.



5. The missing packages should now all be installed now Go to the NS-3 website here <https://www.nsnam.org/releases/ns-3-42/> and download the NS-3 package. Save it to your home folder.

Releases

ns-3.44

ns-3.43

ns-3.42

Authors

Documentation

Download

ns-3.41

ns-3.40

ns-3.39

ns-3.38

ns-3.37

ns-3.36

ns-3.35

ns-3.34

Home > Releases > ns-3.42

ns-3.42

ns-3.42 was released on May 29, 2024, due to contributions from [twenty-four authors](#). The main feature addition was support for the 3GPP 38.311 Non-Terrestrial Networks channel model, including a new circular aperture antenna model described in that channel model, and a mobility model supporting geocentric positions and conversion to topocentric coordinate systems. The LR-WPAN and Wi-Fi modules also made numerous small feature improvements and bug fixes, including a new trace helper for tracing Wi-Fi reception outcomes. Improvements to the TCP BBR and Cubic models were also published, among many additional improvements and bug fixes listed in the [RELEASE_NOTES](#) and [CHANGES](#) files.

Download

The ns-3.42 release download is available from [this link](#). This download is a source archive that contains some additional tools (bake, netanim) in addition to the ns-3.42 source. The ns-3 source code by itself can also be checked out of our Git repository by referencing the tag 'ns-3.42'.

Documentation

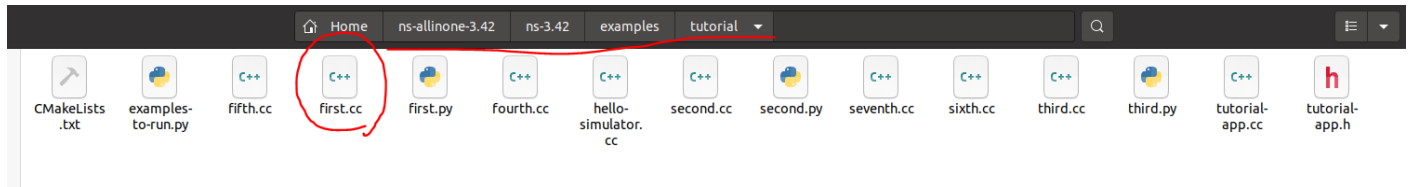
The documentation is available in several formats from [this link](#).

6. Extract the file in the home folder by right-clicking the compressed file and selecting "Extract here". You should now have the ns-allinone-3.42 folder available.
7. Open the terminal and navigate to the ns-allinone-3.42 folder: **cd ns-allinone-3.42/**
8. Enter the command **./build.py --enable-examples --enable-tests** to install NS-3.
9. The installation will take quite some time. I believe mine took 20-ish minutes, but it may take up to an hour to finish installing, so just make sure your VM doesn't fall asleep.
10. After NS-3 is done installing, you want to make sure it is working. Follow this video and jump to 7:00 to see how to test out NS-3:
https://www.youtube.com/watch?v=cPpJ_mJLkzo
 - a. In your terminal, navigate to the ns-3.42 folder: **cd ns-allinone-3.42/ns-3.42/**
 - b. Run the command: **./ns3 run hello-simulator**

Your output should look something like this:

```
kbjackson@kbjackson-VirtualBox: ~/ns-allinone-3.42/ns-3.42
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$ ./ns3 run hello-simulator
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/...utorial/ns3.42-hello-simulator-default
Hello Simulator
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$
```

- c. Now we should test out an example code. Go to your folder and find the tutorials folder. We will be testing out first.cc



- d. Click and copy the first.cc (ctrl+C) and navigate to your scratch folder and paste the copy (ctrl+V) of the file there.



- e. Now go back to your terminal and run the command **./ns3 run scratch/first.**

```
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch/first
-- Using default output directory /home/kbjackson/ns-allinone-3.42/ns-3.42/build
-- Proceeding without cmake-format
-- find_external_library: SQLite3 was found.
-- GSL was found.
-- Precompiled headers were enabled
-- Processing src/antenna
-- Standard library Bessel function has been found
-- Processing src/aodv
-- Processing src/applications
-- Processing src/bridge
-- Processing src/brite
-- Skipping src/brite
-- Processing src/buildings
-- Processing src/click
-- Skipping src/click
-- Processing src/config-store
-- Processing src/core
-- Boost Units have been found.
-- Processing src/csma
```

You may see some other stuff going on if it's the first time running this code, so in the end, you should see this output:

```
[0/2] Re-checking globbed directories...
[2/2] Linking CXX executable ../build/scratch/ns3.42-first-default
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$
```

11. Congrats! You have successfully installed NS-3! Next, we need to make sure the VM has OpenSSL upgraded.

PART 3 - OpenSSL Upgrade:

1. Now the Ubuntu VM should already have an OpenSSL library, so this part of the whole project setup should be very short. To check, go to your terminal and enter **openssl version**. It should give you the version number.

```
kbjackson@kbjackson-VirtualBox:~$ openssl version
OpenSSL 1.1.1f  31 Mar 2020
```

2. To check for upgrades, enter **sudo apt-get install openssl**. You will need to enter your password. You will see if there are any upgrades available.

```
kbjackson@kbjackson-VirtualBox:~$ sudo apt-get install openssl
[sudo] password for kbjackson:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssl is already the newest version (1.1.1f-1ubuntu2.24).
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 libfprint-2-tod1 libfwupdplugin1 libxmlb1
  linux-image-5.15.0-134-generic linux-modules-5.15.0-134-generic
  linux-modules-extra-5.15.0-134-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

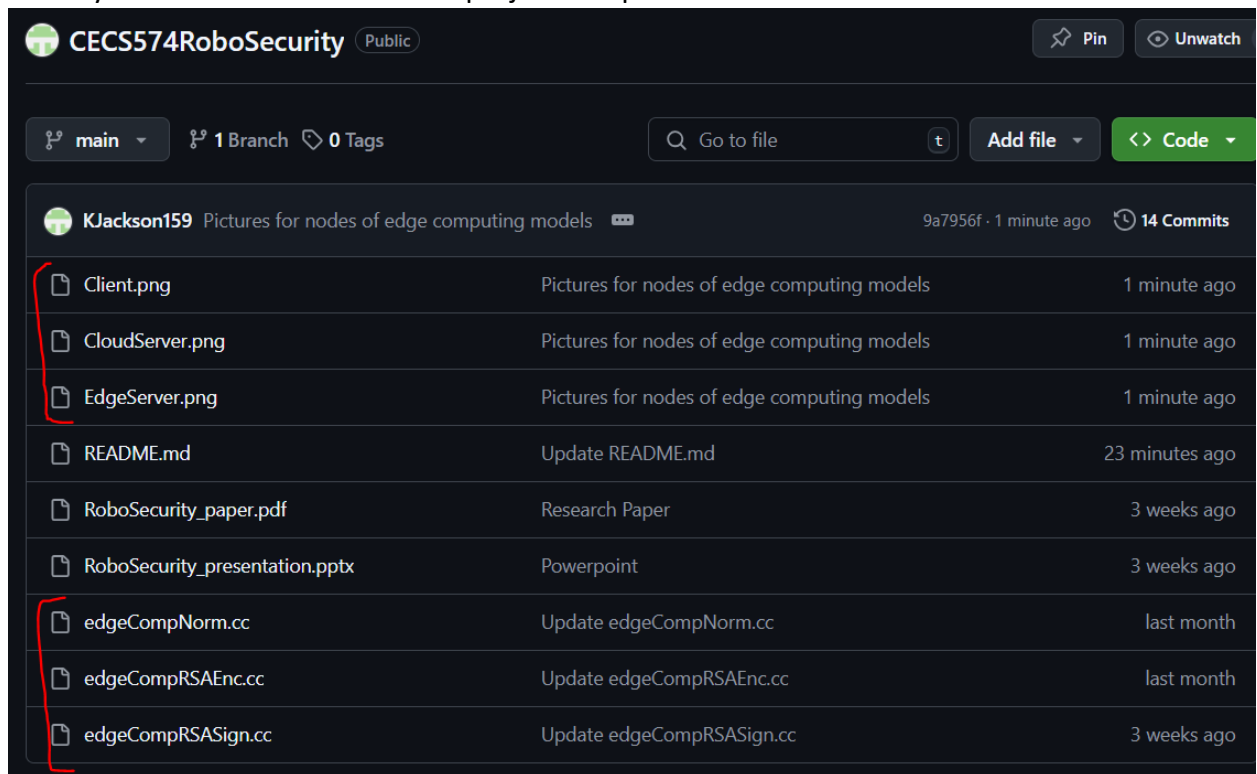
3. If there are upgrades, you can upgrade by entering **sudo apt-get upgrade openssl**.

PART 4 - Download and Run Code (plus OpenSSL setup in NS-3):

1. Before we can run my c++ code that uses OpenSSL, we need to modify a CMakeLists.txt file located in the scratch folder and modify lines 67 and 72 to include OpenSSL for NS-3.

```
67 find_package(OpenSSL) ←
68 build_exec(
69     EXECNAME ${scratch_name}
70     EXECNAME_PREFIX ${target_prefix}
71     SOURCE_FILES "${source_files}"
72     LIBRARIES_TO_LINK "${ns3-libs}" "${ns3-contrib-libs}" "${OPENSSL_LIBRARIES}"
73     EXECUTABLE_DIRECTORY_PATH ${scratch_directory}/
74 )
75 endfunction()
76
```

2. In the VM, open Firefox and go to my GitHub page (<https://github.com/KJackson159/CECS574RoboSecurity>) and download the .cc files. Save them to the **scratch** folder under ns-3.42. Also, download the png images and save them to your **Pictures** folder. The pictures are for the nodes in the NetAnim animation which you will see in PART 5 of the project setup.



3. Open terminal and navigate to the ns-3.42 folder: **cd ns-allinone-3.42/ns-3.42/**

```
kbjackson@kbjackson-VirtualBox:~$ cd ns-allinone-3.42/ns-3.42/
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$
```

4. Run any .cc file by entering `./ns3 run scratch/<FILE NAME>`. See screenshot example below. (If there were any changes made, it may take some time for the executable to finish compiling before it runs. Your computer fan may get loud if there's a lot the executable needs to take account for *like the screenshot below where there's 993 things to link*, so make sure you're not running too many programs on your actual computer as well just to be safe.):

```
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$ ./ns3 run scratch/edgeCompRSASign
[0/2] Re-checking globbed directories...
[993/993] Linking CXX executable ../build/scratch/ns3.42-edgeCompRSASign-default
At time +2e+09ns, 10.1.1.1 sent encrypted msg to 10.1.14.1
At time +2e+09ns, 10.1.2.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.3.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.4.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.5.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.6.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.7.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.8.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.9.1 sent msg to 10.1.13.1
At time +2e+09ns, 10.1.10.1 sent msg to 10.1.13.1
10.1.13.1 received msg from 10.1.2.1 at +2.002e+09ns
Received msg: This is a normal msg from client node 2
At time +2.002e+09ns, 10.1.13.1 sent msg to 10.1.2.1

10.1.13.1 received msg from 10.1.5.1 at +2.002e+09ns
Received msg: This is a normal msg from client node 5
At time +2.002e+09ns, 10.1.13.1 sent msg to 10.1.5.1
```

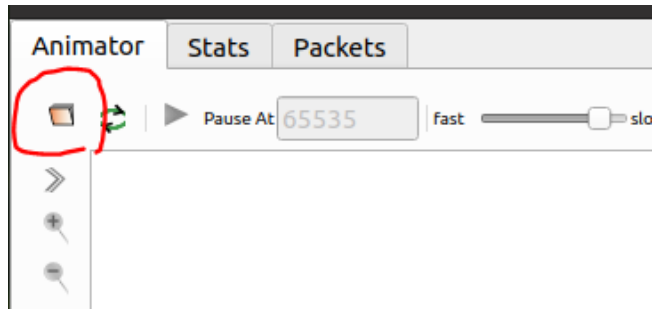
5. Congrats! You successfully ran the c++ code in NS-3!

PART 5 - Run Simulator Animation:

1. To run the NetAnim program, go to your terminal and enter `../netanim-3.109/NetAnim`. This will open a new window of the program, but there will be nothing in it because we haven't selected a model to play the simulation animation yet.

```
kbjackson@kbjackson-VirtualBox:~/ns-allinone-3.42/ns-3.42$ ../netanim-3.109/NetAnim
```

2. Go to the folder icon on the upper left.

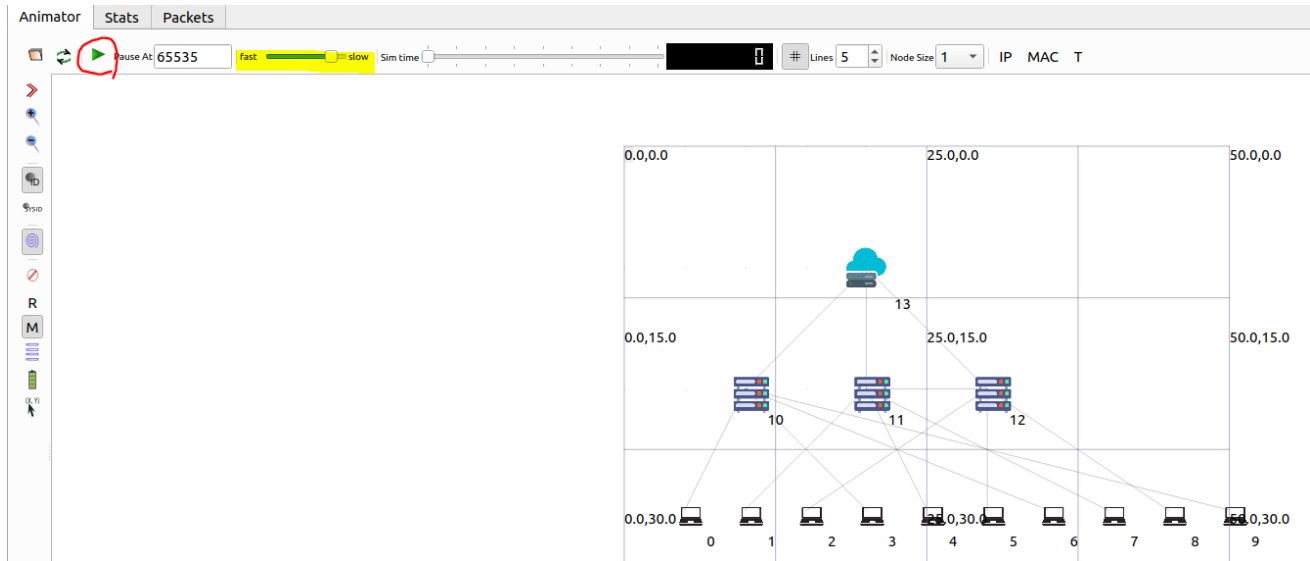


3. Scroll down and select the .xml file that matches the code you ran (for this example, I ran the edgeCompRSASign.cc file, which creates the E2EE.xml file for the animation):

```
474 //Capture all packets/data with Wireshark (see .pcap files)
475 AsciiTraceHelper ascii;
476 pp1.EnableAsciiAll(ascii.CreateFileStream("E2EE.tr"));
477 pp1.EnablePcapAll("E2EE");
478
479 //Simulate Edge Computing model using NetAnim, and configure nodes to specific locations of the graph
480 AnimationInterface anim("E2EE.xml");
481 uint32_t x_pos = 5;
```

kbjackson ns-allinone-3.42 ns-3.42			
Name	Size	Type	Modified
cmake-cache			14:45
contrib			5 Mar
doc			5 Mar
examples			5 Mar
__pycache__			5 Mar
scratch			10 Apr
src			5 Mar
testpy-output			5 Mar
utils			5 Mar
AUTHORS	14.0 kB	Text	29 May 202
CHANGES.md	214.7 kB	Text	29 May 202
CMakeLists.txt	6.0 kB	Text	29 May 202
CONTRIBUTING.md	18.4 kB	Text	29 May 202
DIRIcStats.txt	394 bytes	Text	5 Mar
E2EE.tr	122.3 kB	Text	15:00
E2EE.xml	25.0 kB	Markup	15:00
E2EE-0-1.pcap	3.4 kB	Packet Capture (PCAP)	15:00
E2EE-1-1.pcap	558 bytes	Packet Capture (PCAP)	15:00
E2EE-2-1.pcap	558 bytes	Packet Capture (PCAP)	15:00

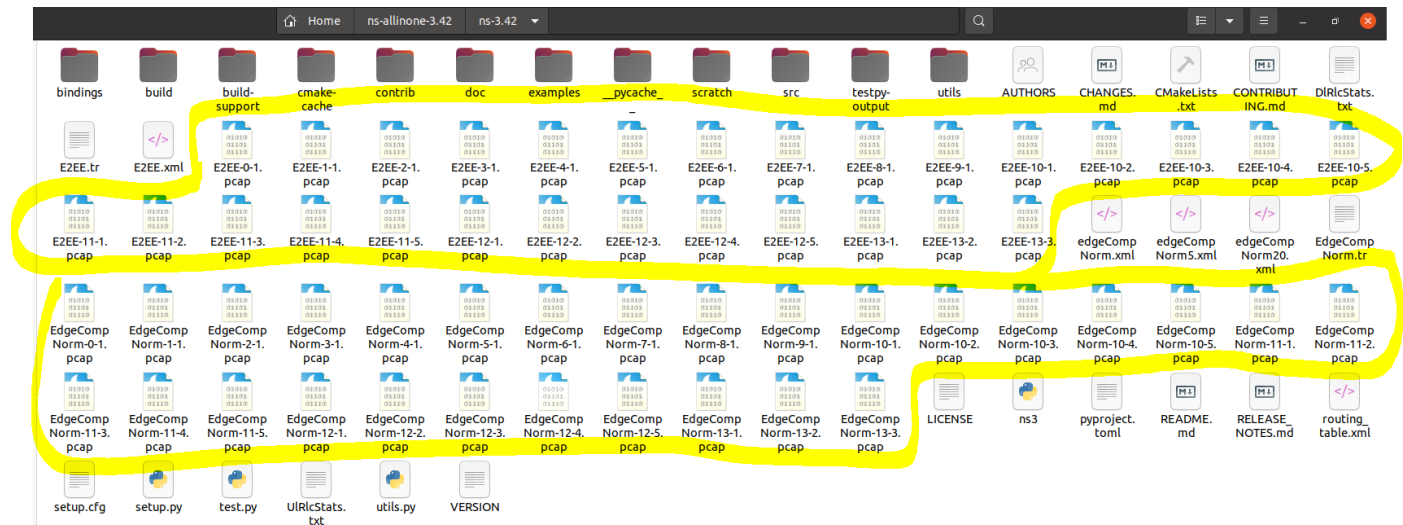
- Once you open the .xml file, you will see the model. To play the animation, click the play button in the upper-left corner, and you should see the visualization of packets traveling to and from each node. It will repeat a pattern 3 times before stopping the animation.



- If you want to see the animation again, click the refresh button (next to the play button), then click play.

PART 6 - Run Wireshark:

1. The .cc files I made includes the creation of .pcap files that capture the traffic in the edge computing models, so all you have to do is go in your files, open the ns-3.42 folder, and select the .pcap file you want to view.



2. Here, you can view the messages that were sent between nodes, and you see who the senders and receivers are for that node.

EdgeCompNorm-13-3.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.1	10.1.14.1	UDP	69	8080 → 8080 Len=39
2	0.004001	10.1.14.1	10.1.1.1	UDP	77	8080 → 8080 Len=47
3	2.000000	10.1.1.1	10.1.14.1	UDP	69	8080 → 8080 Len=39
4	2.004001	10.1.14.1	10.1.1.1	UDP	77	8080 → 8080 Len=47
5	6.000000	10.1.1.1	10.1.14.1	UDP	69	8080 → 8080 Len=39
6	6.004001	10.1.14.1	10.1.1.1	UDP	77	8080 → 8080 Len=47

Frame 1: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)

Point-to-Point Protocol

Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.14.1

User Datagram Protocol, Src Port: 8080, Dst Port: 8080

Data (39 bytes)

```
0000 00 21 45 00 00 43 00 00 00 00 3e 11 00 00 0a 01  !E..C..>.....
0010 01 01 0a 01 0e 01 1f 90 1f 90 00 2f 00 00 54 68  ....../..Th
0020 69 73 20 69 73 20 61 20 6e 6f 72 6d 61 6c 20 6d  is is a normal m
0030 73 67 20 66 72 6f 6d 20 63 6c 69 65 6e 74 20 6e  sg from client n
0040 6f 64 65 20 31                                ode 1
```