OMNeT++ and Hardware-In-The-Loop Simulation Setup Guide

Required Environment:

- Operating System: **Ubuntu 20.04**

OMNeT++ Version:

- Required Version: OMNeT++ 6.0.1
- INET Version: INET 4.4.0
- Download Link: https://omnetpp.org/download/

Installing OMNeT++:

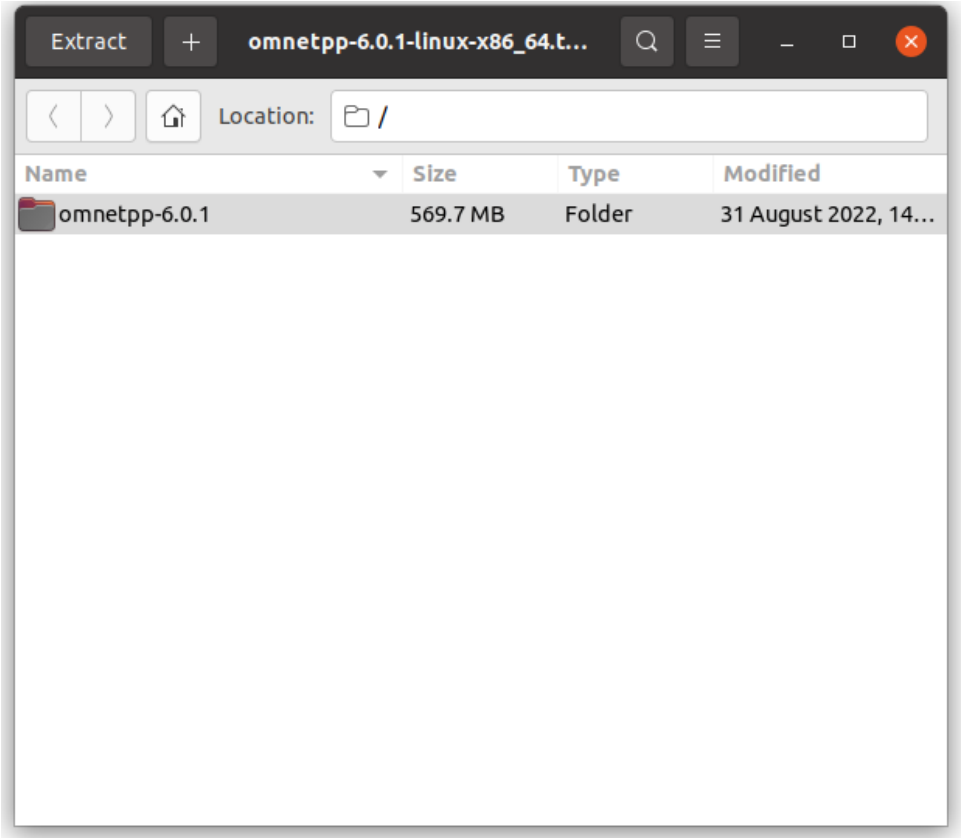
Before we proceed to installing OMNeT++, we need to install perquisites libraries required to run OMNet++. You can either follow instructions here or follow official instructions given by OMNeT++ in Official Guide page 25.

1. Open the terminal in your Ubuntu 20.04.
2. To install the required packages, type in the terminal:

```
sudo apt-get update
sudo apt-get install build-essential clang lld gdb bison flex
sudo apt-get install python3 python3-pip qtbase5-dev qtchooser
sudo apt-get install perl qt5-qmake qtbase5-dev-tools
sudo apt-get install libqt5opengl5-dev libxml2-dev zlib1g-dev
sudo apt-get install doxygen graphviz libwebkit2gtk-4.0-37
sudo apt-get install libavformat-dev mpi-default-dev
sudo apt-get install openscenegraph-plugin-osgearth libosgearth-dev
pip install --user --upgrade numpy pandas matplotlib scipy
pip install seaborn posix_ipc
```

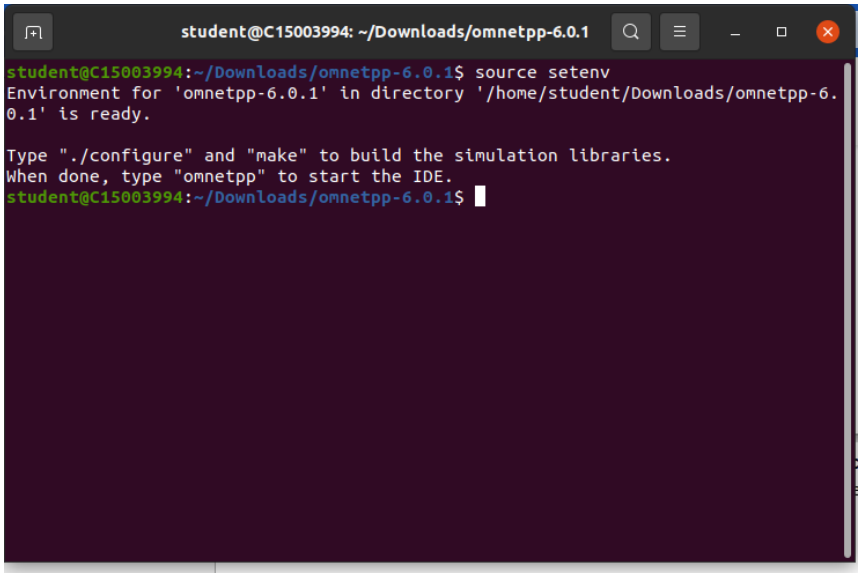
3. After installing packages, download your OMNeT++ 6.0.1 from this link.
4. Go to your downloads, right click on **"omnetpp-6.0.1-linux-x86_64.tgz"** and open with Archive Manager.

5. Click on “Extract” and give it your desired location.



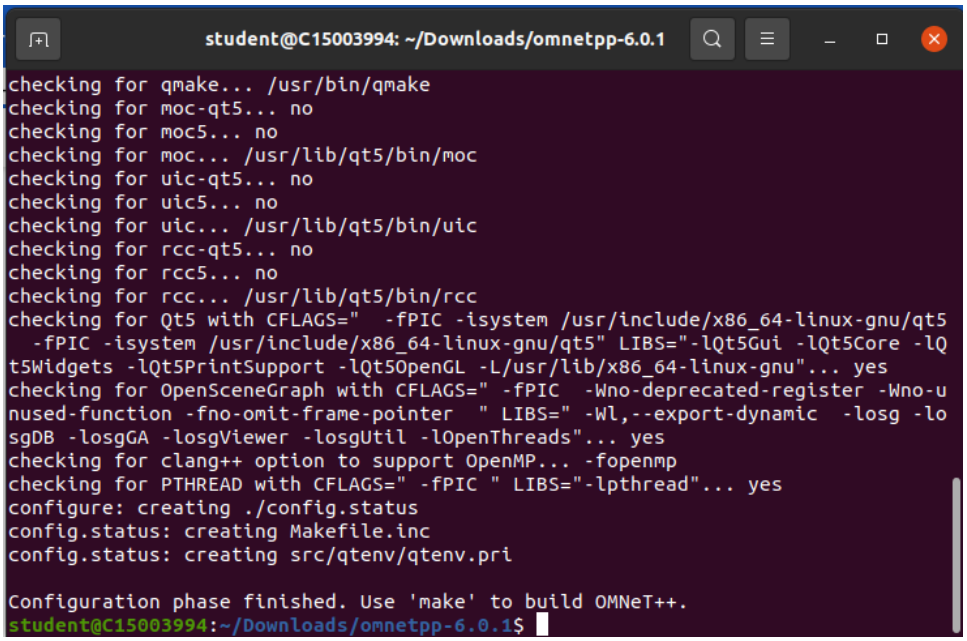
6. Open the extracted folder in the terminal by typing “**cd omnetpp-6.0.1**”.
7. Upon opening extracted folder through terminal, type following commands:

```
source setenv
```

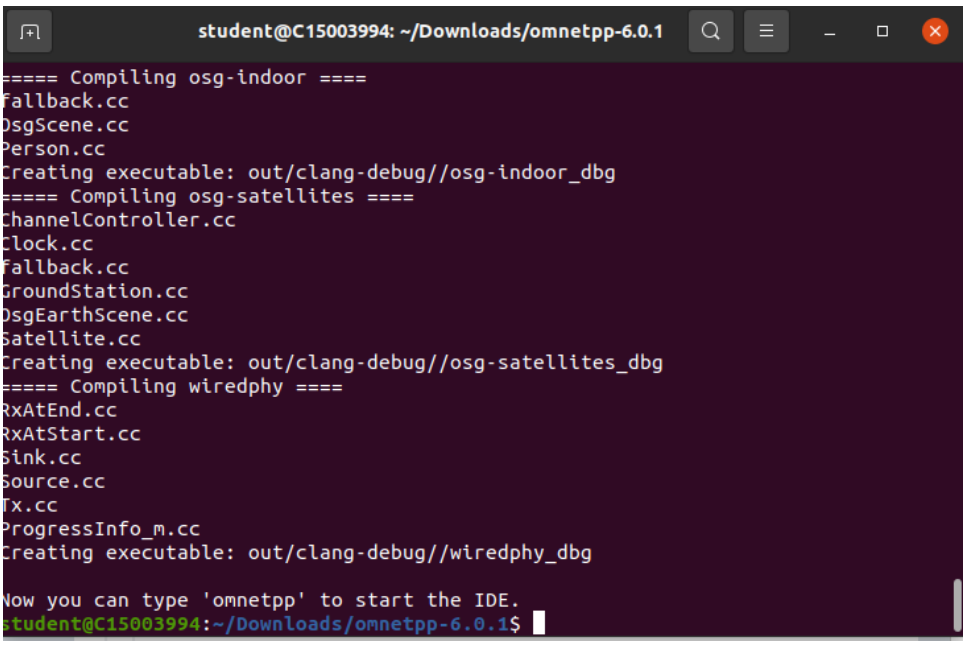


Now type “ ./configure ” and let it run

8. If you have done everything correctly till this point, you should get this message in your terminal



9. Now proceed by typing " make " in your terminal. This can take a while to complete.
10. If you see this, congratulations, your OMNeT++ is ready to use.



Side Note: If **omnetpp** command does not open IDE after a restart, go to installation folder and run **"source setenv"** command and then type omnetpp.

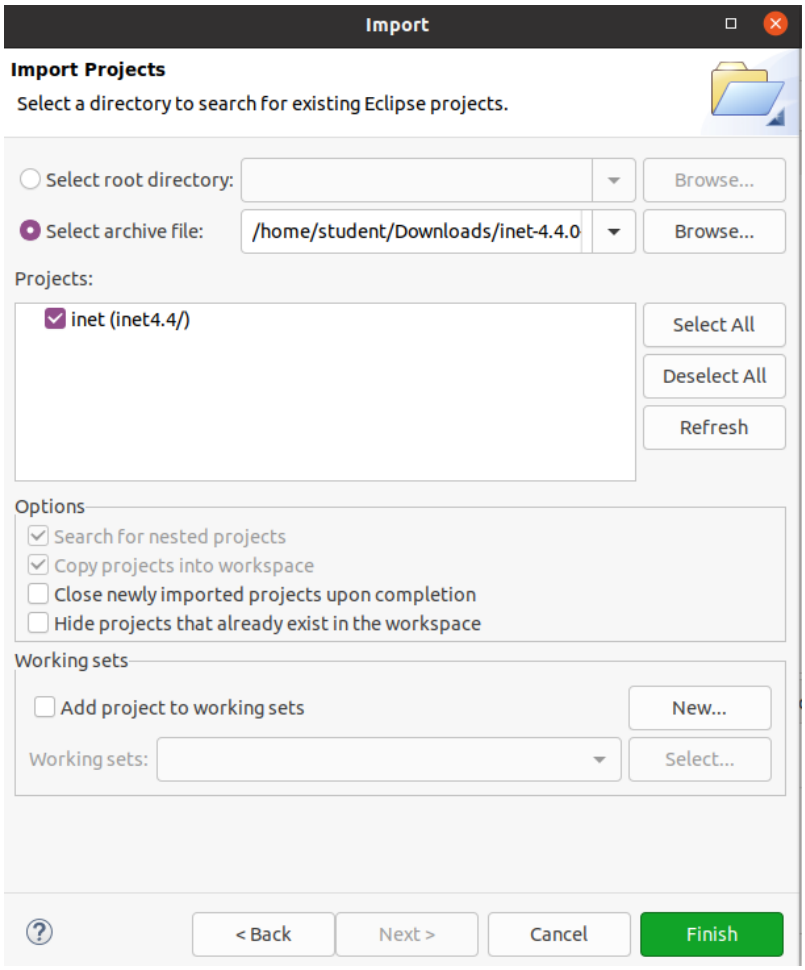
11. Type "**omnetpp**" in your terminal to open the IDE.

12. Once opened, cancel the welcome tab. Now it will ask you to install INET framework and import examples. Select both and click on “OK”.

Now your OMNeT++ is all ready to use.

Installing INET:

1. Download INET 4.4.0 for OMNeT++ 6.0 from the official website.
2. From the menu on top left, click on File -> Import -> Existing Projects into Workspace -> Select archive file.



.

3. Once imported, right click on your inet folder -> Properties -> Expand OMNeT++ -> Project Features -> Enable “Network emulation support showcases” and “Network emulation support examples” and click OK.
4. Right click on INET folder and click on Build Project.

Installing Simu5G:

1. Once building is complete, In the menu bar, click Help | Install Simulation Models... | Select Simu5G 1.2.1 and click Install Project
2. Wait until the building process is complete. This may take up to several minutes.
3. If the method above does not work, please refer to the original Simu5G installation guide to look at other installation methods.
4. If you the name of your inet folder is “inet” and not “inet4.4”, right click on Simu5G directory in project explorer -> Click on Properties -> Project References -> Uncheck “inet4.4” and check “inet” -> Click on Apply and Close.
5. Right click on Simu5G and click on Build Project, it might take few minutes.
6. From project explorer, Simu5G-1.2.1 -> Simulations -> LTE -> Demo -> Right click on omnet.ini and run to verify the installation.

Connecting Simu5G Simulation with External Hosts:

For this simulation, these are IP addresses of video streaming server and client host, this should be adjusted depending upon the case.

Video Streaming Server: 192.168.0.8

Video Streaming Client: 192.168.0.35

To connect the Simu5G with external host, we need to use ExtLowerEthernetInterface that uses sockets to send data into the network. To avoid packet duplication and isolated environment, we will use network namespaces in Linux. Linux network namespaces are a Linux kernel feature allowing us to isolate network environments through virtualization. Namespaces are the basis of container technologies like Docker or Kubernetes. I would recommend watching this video to know more about network interfaces.

1. First of all, we will give OMNeT++ permission to read data from sockets, make sure you run these commands from new terminal.

```
sudo setcap cap_net_raw,cap_net_admin=eip Downloads/omnetpp-6.0.1/bin/opp_run

sudo setcap cap_net_raw,cap_net_admin=eip Downloads/omnetpp-6.0.1/bin/opp_run_dbg

sudo setcap cap_net_raw,cap_net_admin=eip Downloads/omnetpp-6.0.1/bin/opp_run_release
```

2. Now we create namespaces to achieve isolated network for the simulation.

```
sudo ip netns add ns1

sudo ip netns add ns2
```

3. For each interface, we will create a veth pair that will act as a pipe and connect the simulation with our namespace.

```
sudo ip link add veth1 type veth peer name sim-veth1
sudo ip link add veth2 type veth peer name sim-veth2
sudo ip link set veth1 netns ns1
sudo ip link set veth2 netns ns2
sudo ip netns exec ns1 ip link set veth1 up
sudo ip netns exec ns2 ip link set veth2 up
sudo ip link set sim-veth1 up
sudo ip link set sim-veth2 up
```

4. Now, assign IP to our veth interfaces inside our namespace ns1 and ns2.

```
sudo ip netns exec ns1 ip addr add 192.168.2.2/24 dev veth1
sudo ip netns exec ns2 ip addr add 192.168.3.2/24 dev veth2
sudo ip netns exec ns1 ip link set lo up
sudo ip netns exec ns2 ip link set lo up
```

5. To provide internet access to these newly created namespaces, we use macvlan interface as a bridge that will be connected to host ethernet interface. This will allow us to send packets outside directly from our namespace and would allow us to receive packets into our namespace without the help of simulation host.

```
sudo ip link add mvl1 link eno1 type macvlan mode bridge
sudo ip link add mvl2 link eno1 type macvlan mode bridge
sudo ip link set mvl1 netns ns1
sudo ip link set mvl2 netns ns2
sudo ip netns exec ns1 ip link set mvl1 up
sudo ip netns exec ns2 ip link set mvl2 up
```

6. Now we assign IP to these macvlan interfaces that will be used by other hosts to send packets into our namespaces. These IP addresses should be in same subnet as server and client host.

```
sudo ip netns exec ns1 ip addr add 192.168.0.9/24 dev mvl1
sudo ip netns exec ns2 ip addr add 192.168.0.36/24 dev mvl2
```

7. Now we have to add to routes traffic that would send packets to the destination through simulation instead of direct route. These IP addresses should be modified based on scenario.

```
sudo ip netns exec ns1 ip route add 192.168.0.35/32 dev veth1
sudo ip netns exec ns2 ip route add 192.168.0.8/32 dev veth2
```

8. Now add a route in server to destination through simulation, so all the packets destined towards destination will go through the simulation.

   ```
   For Windows: route ADD 192.168.0.35 MASK 255.255.255.0 192.168.0.9
   For Linux: ip route add 192.168.0.35/32 via 192.168.0.9
   ```

9. Now add a route in destination to server through simulation, so all the packets destined towards server will go through the simulation.

   ```
   For Windows: route ADD 192.168.0.8 MASK 255.255.255.0 192.168.0.36
   For Linux: ip route add 192.168.0.8/32 via 192.168.0.36
   ```

10. Disable TCP checksum offloading to make sure that TCP checksum is calculated.

    ```
    sudo ip netns exec ns1 ethtool --offload veth1 rx off tx off
    sudo ip netns exec ns2 ethtool --offload veth1 rx off tx off
    ```

11. After the configuration, download the simulation given along with this guide. Extract the simulation files in the OMNeT/Samples/Simu5G/emulation/ folder.
12. Go to extracted folder, open routing sub-folder. Make sure to set correct IP addresses of server/client in UE.mrt and router.mrt.

```
ifconfig:

# interface to the external client
name: eth0
    inet_addr: 192.168.3.1
    Mask: 255.255.255.0
    MTU: 1500
    Metric: 1
    POINTTOPOINT MULTICAST

name: cellular
    inet_addr: 10.0.0.1
    Mask: 255.255.255.0
    MTU: 1500
    POINTTOPOINT MULTICAST

ifconfigend.

route:
#Destination     Gateway         Genmask          Flags  Metric  Iface
192.168.2.0      *               255.255.255.0    H      0       cellular
192.168.3.0      *               255.255.255.0    H      0       eth0
192.168.0.8      *               255.255.255.255  H      0       cellular  Replace this with your server IP
192.168.0.35     192.168.3.2     255.255.255.255  H      0       eth0  <=  Replace this with client IP
0.0.0.0          *               0.0.0.0          G      0       cellular

routeend.
```

13. Open omnet.ini file and change ue.extHostAddress to your own client IP address.

```
*.ue.numEthInterfaces = 1
*.ue.eth[0].typename = "ExtLowerEthernetInterface"
*.ue.eth[0].device = "sim-veth2"
*.ue.extHostAddress = "192.168.0.35"
*.ue.ipv4.forwarding = true

**.forwarding = true
```

14. Make sure to run the simulation in the **Express mode.**
15. From the video streaming server host, simply stream to the IP address of video streaming client. Since we have added an indirect between server and client through simulation, the UDP packets will go through simulation instead of direct path.

```
cvlc {VideoFileName} --loop --sout
'#transcode{vcodec=h264,vb=3500,width=1920,height=1080,acodec=mp3,ab
=192,channels=2,samplerate=44100,scodec=none}:rtp{mux=ts,dst=192.168
.0.35,port=4004}' &
```

16. Type following command in Client Host to consume the stream.

```
vlc rtp://4004 &
```