This PDF is about OMNeT++: Using Real Applications in a Simulated Network project reproduction of an experience sharing. Many thanks to the Group Leader: **Jingyu Sun** for his help in completing the reproduction of this blog, I couldn't have completed this project by myself without his help.

I hope this PDF can help you save some time, complete the reproduction faster, and understand the logic structure of the whole project.

The Potential errors and the solutions in this project:

1. When downloading and installing Ubuntu, it is best to allocate **100G** of memory to the Ubuntu to avoid not having enough memory to download OMNeT++.
2. When installing **OMNeT++** there may be some other environmental configuration that is missing. This is normal, Google or query ChatGPT can be quickly resolved.
3. If after completing all the configurations, the program still fails to execute and fails to find a file under a certain path and other similar errors, you can try to rebuild the INET file or replace the INET version. It is recommended to be **careful** because INET build takes about half an hour to one and a half hours.
4. The Ubuntu version must be equal to or higher than **Ubuntu 20.04**. The Ubuntu version and the corresponding version of the INET library can be found in the following link:

INET Framework - Download

5. The reference version of the official tutorial is a low version of the INET library, the low version of the INET library does **not** have a complete Example project need to create their own project configuration ini, NED and other files.

As this project needs to use the INET library emulation file, you must use the Linux system under the OMNeT ++.

To build a Linux system, you need a virtual machine and an image of Ubuntu, the Ubuntu version and the version of the INET library can be found in the article in

ZhenHui Yuan's GitHub option on Prof ZhenHuiYuan official website: Zhenhui Yuan

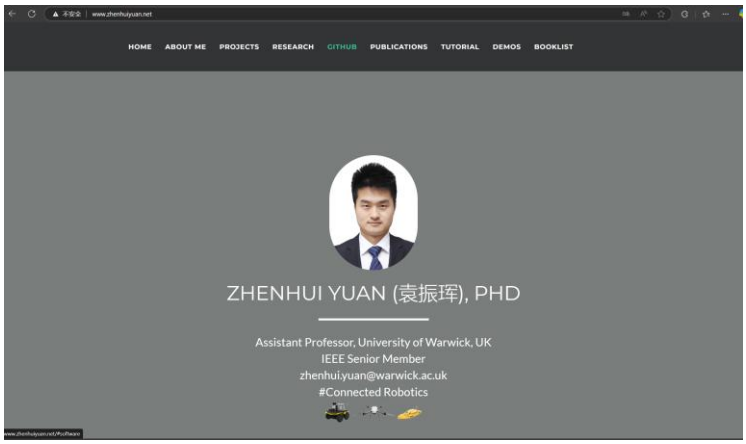


Figure 1

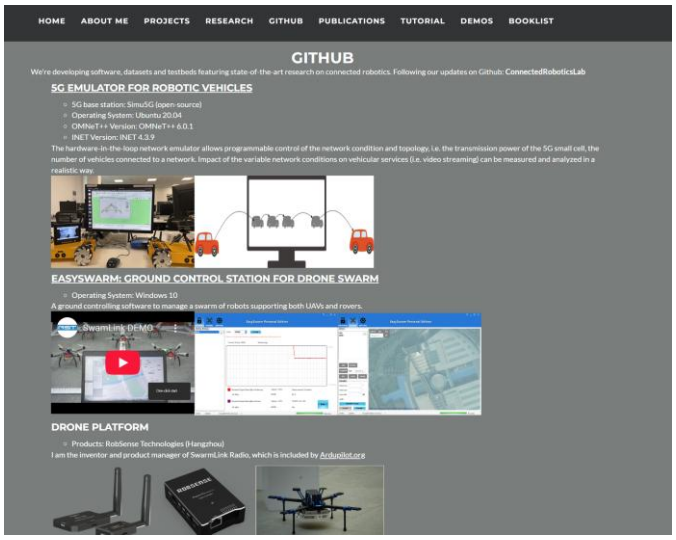


Figure 2

GitHub link to the article:
5G-Emulator-for-robotic-vehicles/5G_HITL.pdf at main · ConnectedRoboticsLab/5G-Emulator-for-robotic-vehicles · GitHub

Step1:

The downloaded version of Ubuntu as well as the INET import tutorial can be fully referenced in the linked tutorial. After completing the tutorial in Figure3, click on the newly imported INET library and click Build to compile the imported INET library (as shown in Figure4).

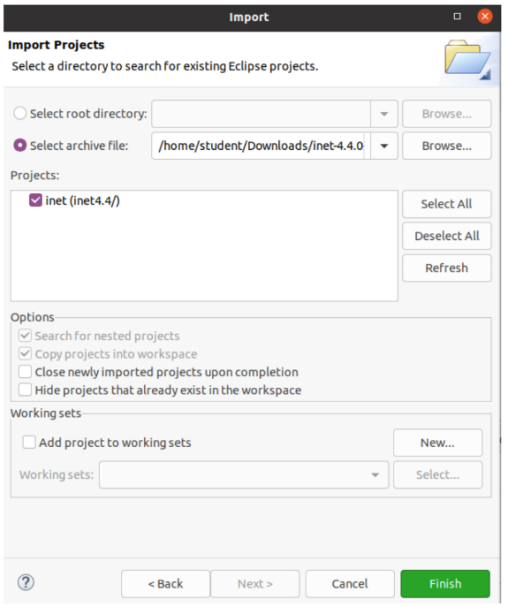
Step2:

Immediately after that, click Import to import the compiled INET library into the

src/example/showcase/emulation/videostreaming path as shown in Figure5. The project under this path is the official compiled and configured project, and you can go to the official tutorial to start the project.

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.

Figure 3

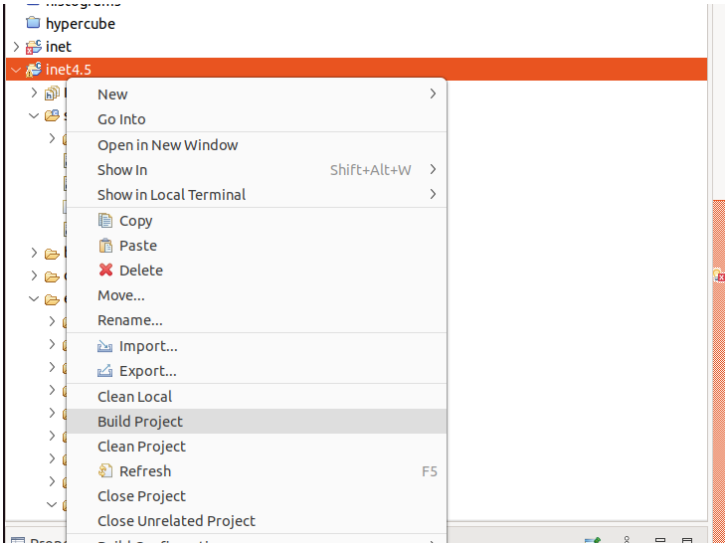


Figure 4

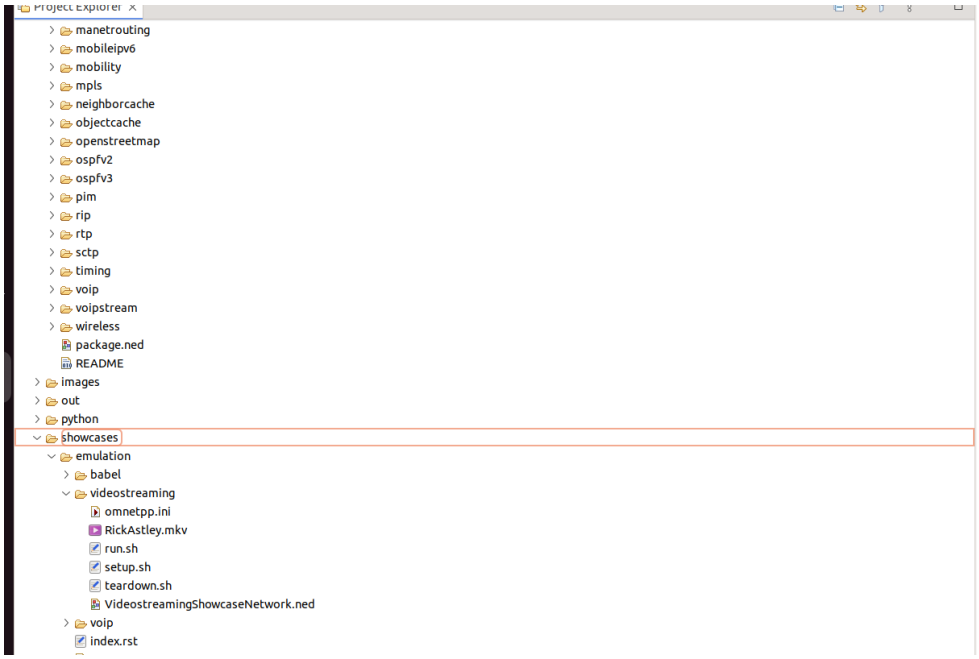


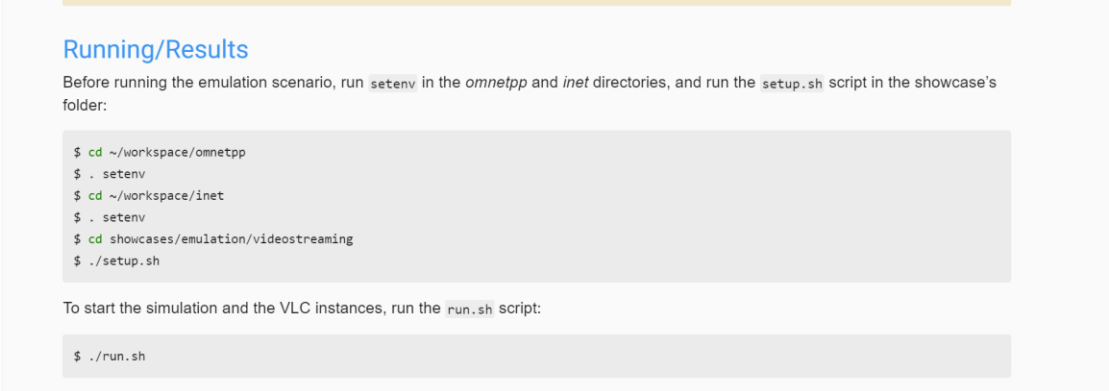
Figure 5

The following is the whole project code introduction, logical framework introduction, read the following and the official link tutorial can help you understand how to build and how to continue to optimize: the following link is the official tutorial.

In the simulation part, mainly references the Using Real Applications in a Simulated Network(Using Real Applications in a Simulated Network — INET 4.5.4 documentation).

Step3:

The first time you build this project you can just follow the commands in Figure6 to start the project directly.



Running/Results

Before running the emulation scenario, run `setenv` in the *omnetpp* and *inet* directories, and run the `setup.sh` script in the showcase's folder:

```
$ cd ~/workspace/omnetpp
$ . setenv
$ cd ~/workspace/inet
$ . setenv
$ cd showcases/emulation/videostreaming
$ ./setup.sh
```

To start the simulation and the VLC instances, run the `run.sh` script:

```
$ ./run.sh
```

Figure 6

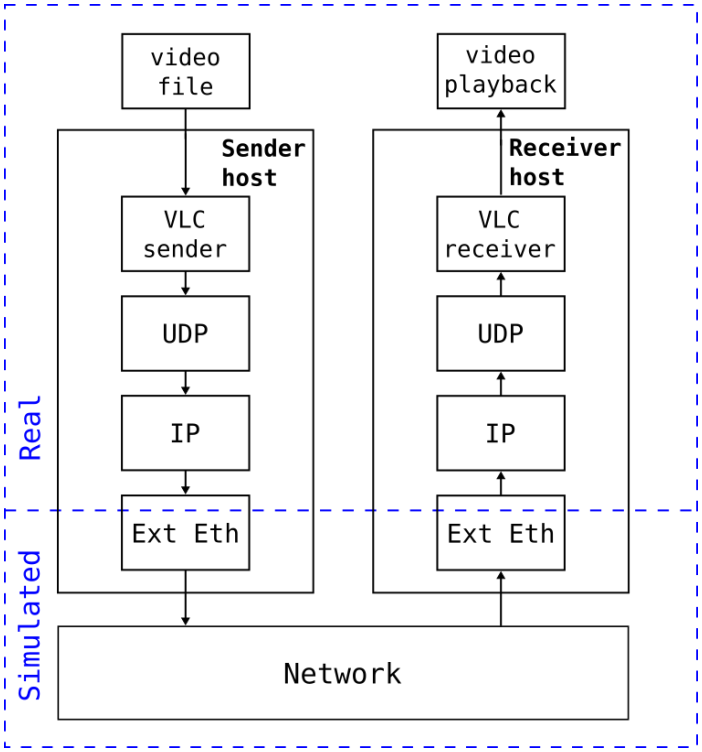


Figure 7 Schematic diagram of project (Omnetpp.org, 2025))

The implementation of the whole project is based on the flow schematic of Figure3. The video transmission between two hosts is implemented through OMNeT ++. The key to this project is to implement the interface between the simulated network and the real network using ExtUpperEthernetInterface from the emulation library of the INET library. Simulate the video transmission in real situation. We will analyse the video quality of the input and output video to study the impact of network protocols, video transcoder on the quality of video transmission and how to optimise the video transmission in the future.

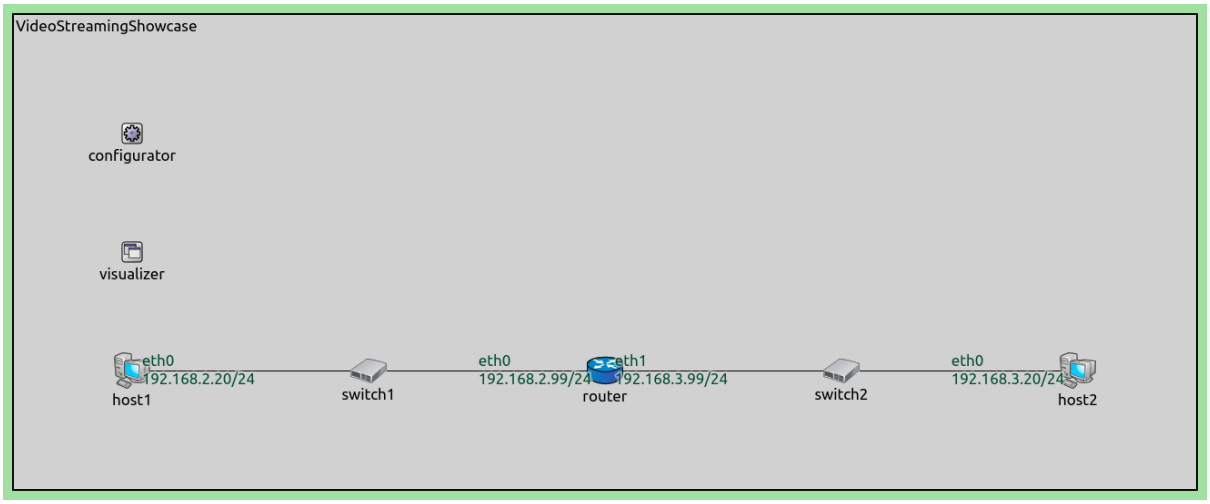


Figure 8 NED network model

Network Construction:

In this project the NED file has created a network called VideoStreamingShowcase, which contains two hosts, two switches and one router.

The **configurator** module in the top of the picture above is an instance of Ipv4NetworkConfigurator which is used to allocate the IP address to each port.

Switch1 and switch2 are instance of EthernetSwitch, which work on the data link layer.

A switch is a key device in the **Data Link** Layer. It uses **MAC** addresses to forward data frames to the correct device within a network. Works in local area networks (LANs) to connect multiple devices (Geeks for Geeks, 2022).

Router is an instance of Router, which works in the **network layer** using Ip address to transmit the data, so that the host1 and host2 can communicate with each other.

```
connections:
    host1.ethg++ <--> Eth100M <--> switch1.ethg++;
    switch1.ethg++ <--> Eth100M <--> router.ethg++;
    router.ethg++ <--> Eth100M <--> switch2.ethg++;
    switch2.ethg++ <--> Eth100M <--> host2.ethg++;
```

Figure 9 code of host1 and host2's internet setup connection

The codes above illustrate how host1 and host2 connect to switches and router by ethernet to simulate the real-world LAN. ethg++ will automatically allocates the port number to each device.

IP setup and transmission principle:

After completing the NED network build setup. The next step is to build the **ini** file to write specific implementation of the network.

```
*.host1.eth[0].typename = "ExtUpperEthernetInterface"
*.host1.eth[0].device = "tapa"
*.host1.eth[0].copyConfiguration = "copyFromExt"

*.host2.eth[0].typename = "ExtUpperEthernetInterface"
*.host2.eth[0].device = "tapb"
*.host2.eth[0].copyConfiguration = "copyFromExt"
```

Figure 10 ini. file host network setup

Set the TypeName of eth [0] and eth [1] to ExtUpperEthernetInterface, which is inherited from the emulation file and is used to connect the real-world interface to the virtual interface. The two interfaces are set to tapa and tapb. The implementation is in setup.sh, which creates tapa and tapb and sets the IP addresses for the interfaces.

```
1   # create TAP interfaces
2   sudo ip tuntap add mode tap dev tapa
3   sudo ip tuntap add mode tap dev tapb
4
5   # assign IP addresses to interfaces
6   sudo ip addr add 192.168.2.20/24 dev tapa
7   sudo ip addr add 192.168.3.20/24 dev tapb
8
9   # bring up all interfaces
10  sudo ip link set dev tapa up
11  sudo ip link set dev tapb up
```

Figure 11 setup.sh: define the tapa and tapb for ini. file

```
*.configurator.config = xml("<config> \
                            <interface hosts='router' names='eth0' address='192.168.2.99' netmask='255.255.255.0'/> \
                            <interface hosts='router' names='eth1' address='192.168.3.99' netmask='255.255.255.0'/> \
                          </config>")
*.router.ipv4.natTable.config = xml("<config> \
                                      <entry type='prerouting' \
                                      packetFilter='has(Ipv4Header) &amp;&amp; Ipv4Header.destAddress.str() == \"192.168.2.99\"' \
                                      srcAddress='192.168.3.99' destAddress='192.168.3.20'/> \
                                    </config>")
```

Figure 12 IP setup to port of router and principle of IP destination transform

The following scripts in Figure8 set the IP address to eth0 and eth1around the **s** to

'192.168.2.99' and '192.168.3.99' in sequence, *. router. ipv4.natTable.config enforces the NAT (Network Address Translation) rule. If the packet destination IP is 192.168.2.99, modify the destination address to 192.168.3.20. Ensure that data from the incoming router at 192.168.2.99 (host1) is routed to 192.168.3.20 (host2).

Video transimission:

After finishing all the network setup, the next step is configuring the VLC video sending and receiving. In this project there is a run.sh file.

```
# start streaming server
cvlc RickAstley.mkv --loop --sout '#transcode{vcodec=h264,acodec=mpga,vb=125k,ab=64k,deinterlace,scale=0.25,threads=2}:rtp{mux=ts,dst=192.168.2.99,port=4004}' &

# start streaming client
vlc rtp://192.168.3.20:4004 &

# start simulation
inet -u Cmdenv -f omnetpp.ini

# kill child processes
trap 'kill $(jobs -pr)' SIGINT SIGTERM EXIT
```

Figure 13 VLC video setup:

The local VLC video has been decoder and transcode to RTP principle. The video will be sent to dst=192.168.2.99, according to the *. router. ipv4.natTable.config configuration in omnetpp.ini file, the video will be sent from 192.168.2.99 to 192.168.3.20. Then start the streaming client and simulation. The whole implementation of this project is finished.

```
$ cd ~/workspace/omnetpp
$ . setenv
$ cd ~/workspace/inet
$ . setenv
$ cd showcases/emulation/videostreaming
$ ./setup.sh
```

To start the simulation and the VLC instances, run the `run.sh` script:

```
$ ./run.sh
```

Figure 14 The comments used to start the project in terminal

After all the implementation setups run the following comments to start the project, the

final VLC video that has been sent to host2 is shown below.

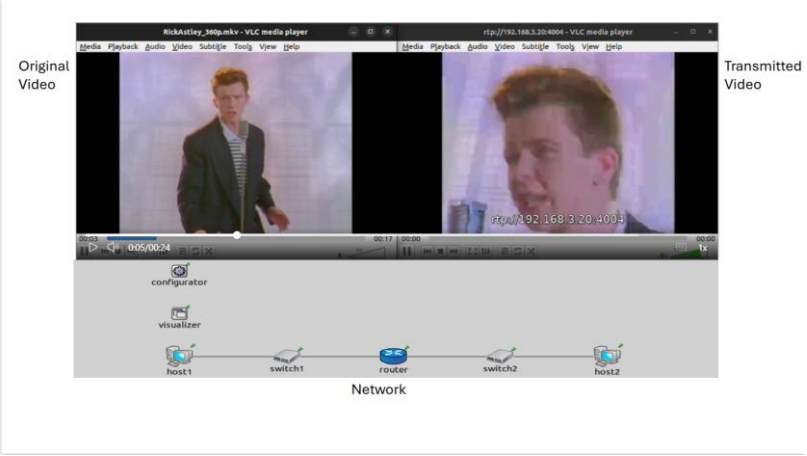


Figure 15 The output video

Comparing the original video with the output video, the quality of the output video has dropped very much. It can be concluded that there is a serious loss of video during transmission.