



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Installation Manual for NS-3 on Ubuntu 20.04 LTS

Prof. Fan Li (Instructor)

Prof. Kashif Sharif (Instructor)

Karim Md Monjurul (TA)

2021.4.8

- Virtualized Instances of Ubuntu or Any Linux-based Distros
 - VMware [Player](#) or [Workstation](#) (16.0 or higher)
 - [VirtualBox](#) (6.1 or higher) and [Extension Pack](#)
 - [Docker](#) Desktop for Windows
 - [WSL or WSL-2 \(Overview\)](#)
 - Minimum Requirements
 - 20GB of Memory Allocation
 - 2-4 GB RAM allocation (less means slower)
 - Specific Windows Build 2004 or higher (for WSL-2)
- Basic Linux Commands (i.e., sudo, apt, ls, cat, nano, cp, mv)
- [Visual Studio Code](#) (Writing Codes)
- Terminal App (Compiling the Codes and Simulator Execution)
- [Wireshark](#) (Packet Sniffer and Analyzer)
- [Gnuplot](#) or Matplotlib (Plotting Graphs)

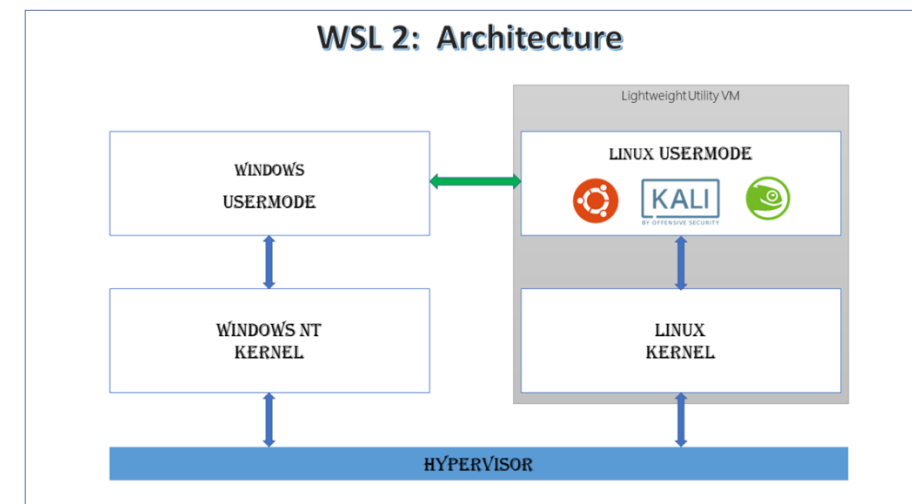
A Short Overview on WSL

Windows Subsystem for Linux

- Allows to install a Linux distribution as an app from the Windows store.
- Execute from a command prompt or PowerShell terminal
- Run Bash shell scripts and GNU/Linux command-line applications:
 - Languages: C, **C++**, Python, Java, GO, NodeJS, etc.
 - Services: Apache, MySQL, MongoDB, etc.

What WSL-2 brings compared to WSL-1

- WSL2 runs on top of the Windows Hypervisor, which is a bare metal **hypervisor**
- Supports memory reclaim (uses only the right amount of **RAM** required for running the **Linux kernel**)
- Better integration with Windows OS



/Prerequisites and Installation Steps to WSL 2

Windows 10 build 18917 or higher.

- To find your Windows version, open Settings>System>About and look for the "OS build" field. os_build.
- **Step-1:** Enable the "Virtual Machine Platform" and "Windows Subsystem for Linux" feature; Alternatively: **Open PowerShell as Administrator and Run:**

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

- **Step-2:** Enable Virtual Machine feature before Ubuntu installation.
 - Require **virtualization** capabilities to use this feature
 - In some cases, you have to enable from **BIOS**.

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

- **Step-3:** Download the Linux kernel update package ([Link](#))
- **Step-4:** Set WSL 2 as your default version

```
wsl --set-default-version 2
```

- **Step-5:** Install your Linux distribution of choice ([Microsoft Store](#))
- **Step-6:** Create a user account and password for your new Linux distribution

- **Step-7:** Check the Distro and WSL version

```
wsl -l -v
```

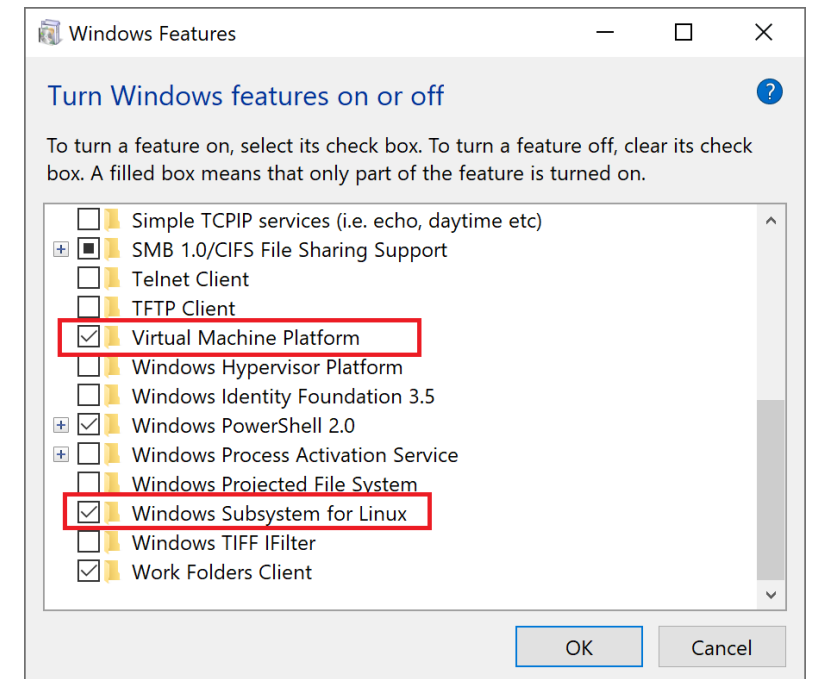
Windows specifications

Edition	Windows 10 Home Insider Preview
Version	1903
Installed on	7/13/2019
OS build	18936.1000
Serial number	

[Change product key or upgrade your edition of Windows](#)

[Read the Microsoft Services Agreement that applies to our services](#)

[Read the Microsoft Software License Terms](#)



Prerequisites and Installation Steps to NS-3 on Ubuntu 20.04

Explain Each Steps and Commands

- **Step-1:** Change the Software Repository (Tsinghua, Aliyun, USTC)

```
sudo sed -i 's#archive.ubuntu.com#mirrors.tuna.tsinghua.edu.cn#g' /etc/apt/sources.list
```

- **Step-2:** Update the Repo and Upgrade the System

```
sudo apt update && sudo apt -y upgrade
```

- **Step-3:** Install Desktop Environment (KDE, XFCE, LXDE, GNOME 3)

```
sudo apt install xfce4 xfce4-goodies
```

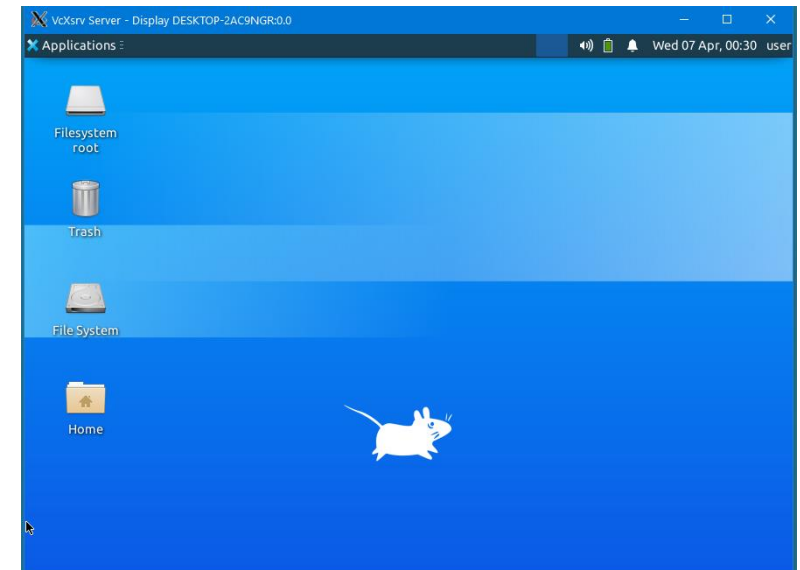
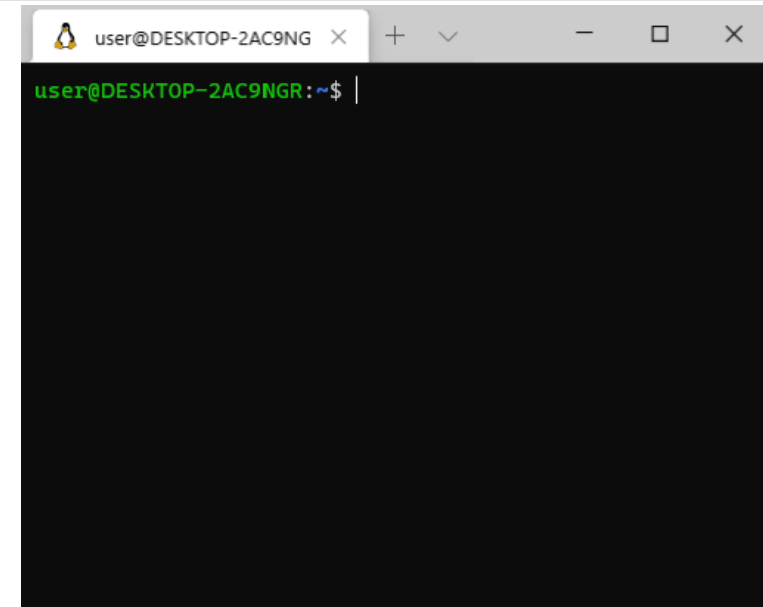
- **Step-4:** Install Core Dependencies

```
sudo apt install build-essential libsqlite3-dev libboost-all-dev libssl-dev git python3-setuptools castxml
```

- **Step-5:** Dependencies for NS-3 Python bindings

```
sudo apt install gir1.2-gocanvas-2.0 gir1.2-gtk-3.0 libgirepository1.0-dev python3-dev python3-gi python3-gi-cairo python3-pip python3-pygraphviz python3-pygccxml
```

```
sudo pip3 install kiwi
```



Explanation to Each Steps to Installation Process

➤ Step-6: ns-3 Specific Dependencies Libraries

```
sudo apt install g++ pkg-config sqlite3 qt5-default mercurial  
ipython3 openmpi-bin openmpi-common openmpi-doc libopenmpi-dev  
autoconf cvs bzip2 unrar gdb valgrind uncrustify doxygen graphviz  
imagemagick python3-sphinx dia tcpdump libxml2 libxml2-dev cmake  
libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake
```

➤ Step-6: Download and Extract ns-3 Install Pack

```
cd  
wget -c https://www.nsnam.org/releases/ns-allinone-3.33.tar.bz2  
tar -xvjf ns-allinone-3.33.tar.bz2
```

➤ Step-7: Install ns-3 Simulator with waf command

```
cd ns-allinone-3.33/ns-3.33/  
./waf configure --enable-examples  
./waf  
cd
```

➤ Step-7.1: Alternatively we can use build.py to compile and build ns-3

```
cd ns-allinone-3.33/  
./build.py --enable-examples --enable-tests
```

ns-3 Package Overview

What NS-3 Pack Includes:

- Directories
 - bake
 - netanim-3.108
 - ns-3.33
 - pybindgen-0.21.0...
- Files
 - build.py, constants.py, util.py

Confirm the Procedure (Terminal):

- Most modules should be built except
 - brite
 - click
 - openflow
- Others should be built including
 - visualizer

Validate Ns-3 Installation

➤ Step-8: Check ns-3 installation

```
cd ns-allinone-3.33/ns-3.33/  
./waf --run hello-simulator
```

The terminal should output

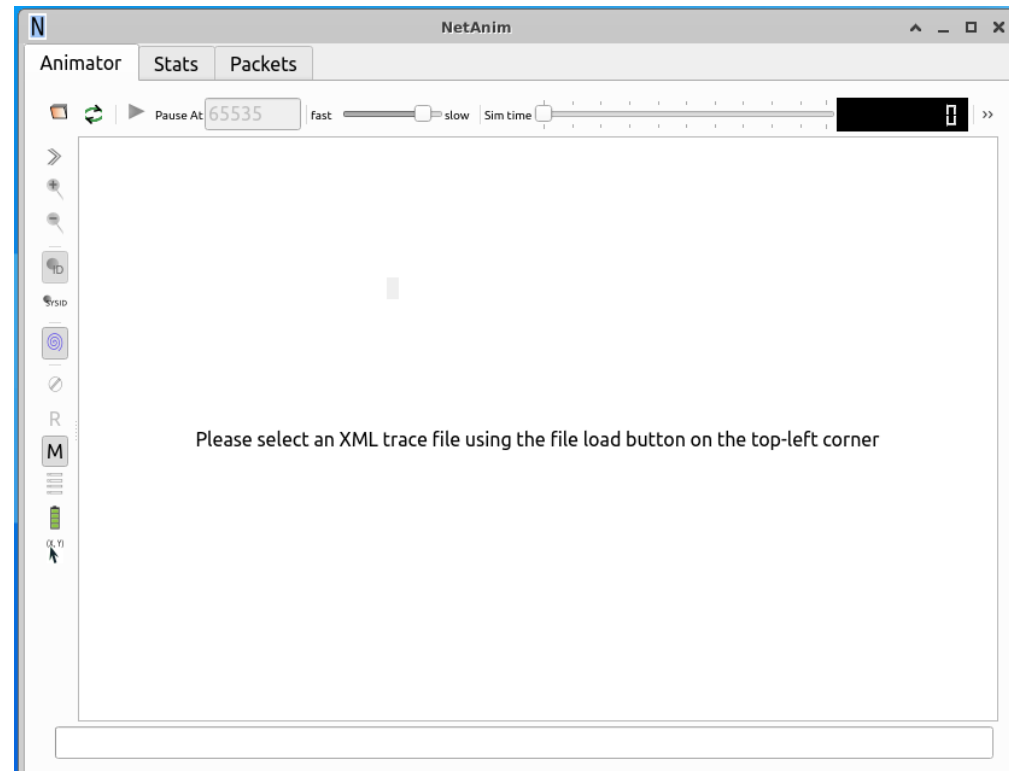
Hello Simulator

```
Modules built:  
antenna                aodv                applications  
bridge                 buildings           config-store  
core                   csma                csma-layout  
dsvd                   dsr                 energy  
fd-net-device          flow-monitor        internet  
internet-apps          lr-wpan             lte  
mesh                   mobility            netanim  
network                nix-vector-routing olsr  
point-to-point         point-to-point-layout propagation  
sixlowpan              spectrum            stats  
tap-bridge             test (no Python)   topology-read  
traffic-control        uan                 virtual-net-device  
visualizer             wave                wifi  
wimax  
  
Modules not built (see ns-3 tutorial for explanation):  
brite                  click                dpdk-net-device  
mpi                    openflow  
  
user@DESKTOP-2AC9NGR:~/ns-allinone-3.33/ns-3.33$ ./waf --run hello-simulator  
Waf: Entering directory `/home/user/ns-allinone-3.33/ns-3.33/build'  
Waf: Leaving directory `/home/user/ns-allinone-3.33/ns-3.33/build'  
Build commands will be stored in build/compile_commands.json  
'build' finished successfully (1.160s)  
Hello Simulator
```

Validate NetAnim Installation

➤ Step-9: Build and Compile netanim-3

```
cd ns-allinone-3.33/netanim-3.108/  
make clean  
qmake NetAnim.pro  
make  
./NetAnim  
cd
```



Using NS-3 Simulator to Build, Run Simulation Scenarios

Compiling examples and custom-written scenarios

➤ Test Scenario

```
./waf --run first
```

```
Waf: Entering directory `/home/user/ns-allinone-3.33/ns-3.33/build'
Waf: Leaving directory `/home/user/ns-allinone-3.33/ns-3.33/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.088s)
ExampleFunction received event at 10s
RandomFunction received event at 18.1653s
Member method received event at 20s started at 10s
```

```
tree examples/tutorial/
```

```
— examples-to-run.py
— fifth.cc
— first.cc
— first.py
— fourth.cc
— hello-simulator.cc
— second.cc
— second.py
— seventh.cc
— sixth.cc
— third.cc
— third.py
— wscript
```

```
ls -l examples/tutorial/
```

```
total 76
-rw-r--r-- 1 user user 859 Jan 10 02:19 examples-to-run.py
-rw-r--r-- 1 user user 6487 Jan 10 02:19 fifth.cc
-rw-r--r-- 1 user user 2464 Jan 10 02:19 first.cc
-rw-r--r-- 1 user user 2238 Jan 10 02:19 first.py
-rw-r--r-- 1 user user 1791 Jan 10 02:19 fourth.cc
-rw-r--r-- 1 user user 894 Jan 10 02:19 hello-simulator.cc
-rw-r--r-- 1 user user 3592 Jan 10 02:19 second.cc
-rw-r--r-- 1 user user 3431 Jan 10 02:19 second.py
-rw-r--r-- 1 user user 10001 Jan 10 02:19 seventh.cc
-rw-r--r-- 1 user user 7252 Apr 7 03:25 sixth.cc
-rw-r--r-- 1 user user 6048 Jan 10 02:19 third.cc
-rw-r--r-- 1 user user 5649 Jan 10 02:19 third.py
-rw-r--r-- 1 user user 1417 Jan 10 02:19 wscript
```

Custom Scenario

```
nano scratch/1.cc
```

```
./waf
```

```
./waf --run scratch/1
```

```
GNU nano 4.8
/*
TCPTestRouteMod v0.1
Two nodes communicating over PPP with TCP protocol
There is a routing node in the middle.
*/

#include "ns3/netanim-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;
Ptr<OutputStreamWrapper> cWndStream;

NS_LOG_COMPONENT_DEFINE ("TCPTest");

class TestApp : public Application{
public:
    TestApp() : m_socket (0),
                m_peer (),
                m_packetSize (0),
                m_nPackets (0),
                m_dataRate (0),
                m_sendEvent (),
                m_running (false),
                m_packetsSent (0) {
    }
    ~TestApp() {
        m_socket = 0;
    }
};
```


Visualize Simulation Scenario using PyViz

PyViz Intro and Configuration in Custom Scenario

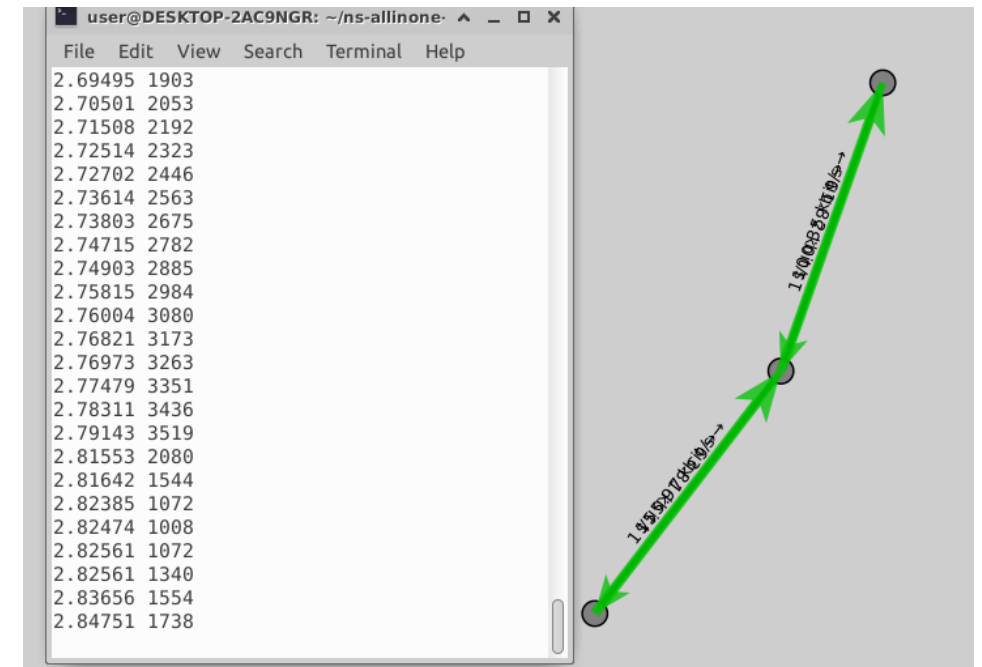
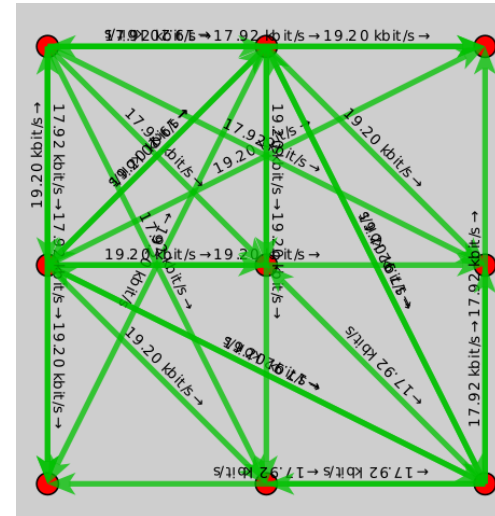
Run Example Code

```
./waf --pyrun src/flow-monitor/examples/wifi-olsr-flowmon.py --vis
```

Make Changes on the Scenario

```
//int main ()  
int main(int argc, char* argv[])  
{  
    Time::SetResolution(Time::NS);  
    LogComponentEnable("TCPTest", LOG_LEVEL_INFO);
```

```
// Read optional command-line parameters (e.g., en  
CommandLine cmd;  
cmd.Parse(argc, argv);  
  
//Creating 3 nodes. 2 will be source dest pair, th  
NS_LOG_INFO("Creating Nodes");  
NodeContainer nodes;  
nodes.Create(3);
```



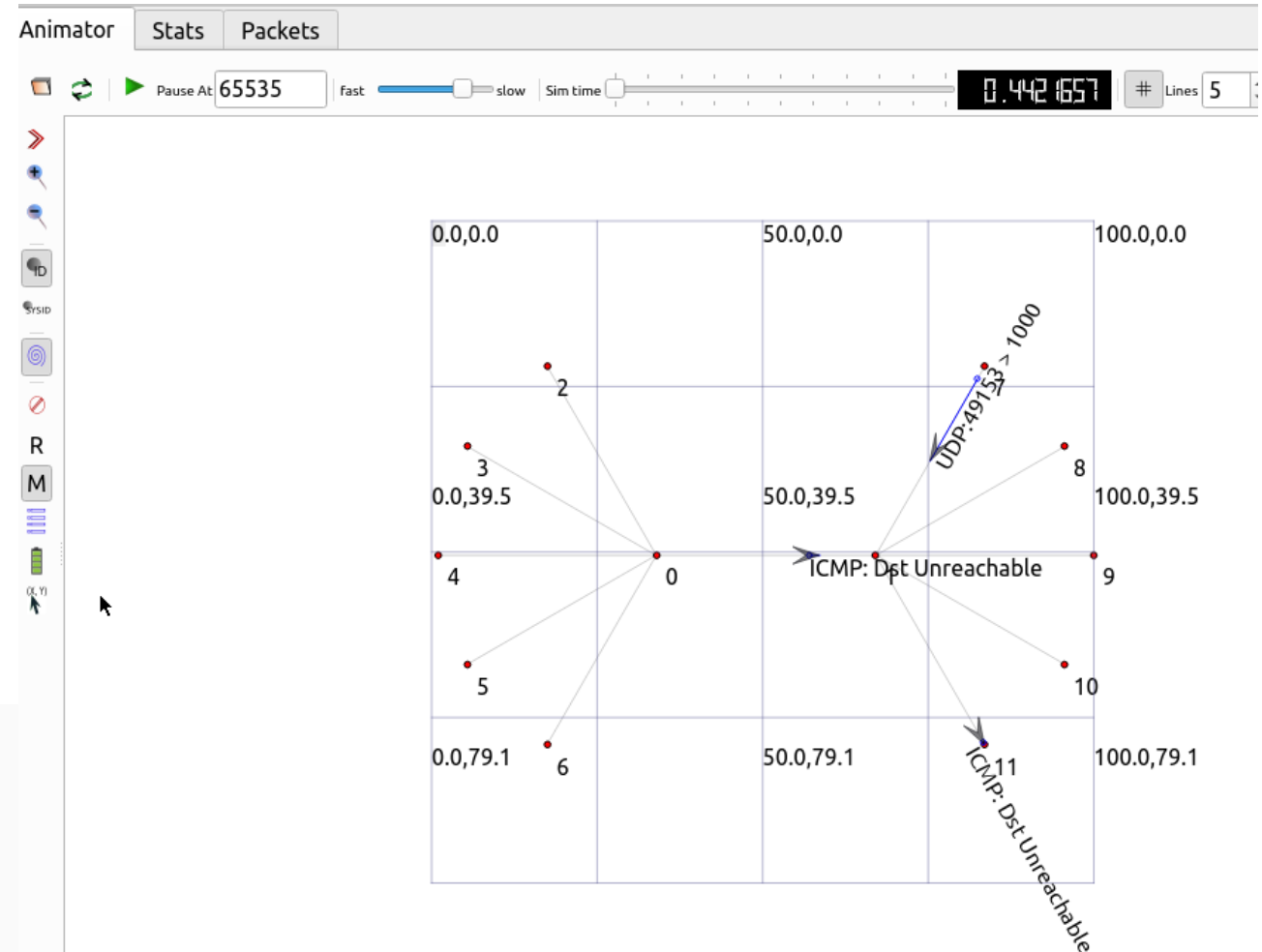
Visualize Simulation Scenario using NetAnim

Enabling NetAnim xml in Simulation

- Add the header file
- Add the .xml output file
- .xml file needs to be open in NetAnim

```
17 #include "ns3/netanim-module.h"  
18 #include "ns3/core-module.h"  
19 #include "ns3/network-module.h"  
20 #include "ns3/internet-module.h"  
21 #include "ns3/point-to-point-module.h"  
22 #include "ns3/applications-module.h"  
23  
24 NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");  
25  
26 using namespace ns3;  
27 Ptr<OutputStreamWrapper> cWndStream;  
28 Ptr<OutputStreamWrapper> ssThreshStream;
```

```
154 sourceApp->SetStartTime(Seconds(1.0));  
155 sourceApp->SetStopTime(Seconds(20.0));  
156  
157 AnimationInterface anim("scratch/first.xml");  
158 anim.SetConstantPosition(nodes.Get(0), 0.0, 0.0);  
159 anim.SetConstantPosition(nodes.Get(1), 20.0, 20.0);  
160  
161 p2p.EnablePcapAll("scratch/TCPTest");  
162  
163 //Initializing the cWndStream
```



Analyzing Packets in Wireshark

Enable Pcap Tracing into the Scenario

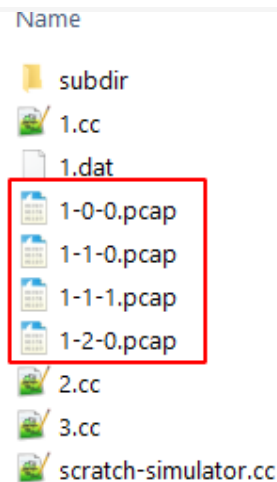
- Add EnablePcapAll function
- Run the simulation
- *.pcap files will be generated
- Two choices to view the pcap files in WSL
 - Install Wireshark on Host Windows
 - Add Wireshark Program Folder to **Environment variables -> Path**

```
explorer.exe .
wireshark.exe scratch/1-0-0.pcap
```

```
//Enabling Pcap Tracing
p2p.EnablePcapAll("scratch/1");

//Initializing the cWndStream
AsciiTraceHelper asciiTraceHelper;
cWndStream = asciiTraceHelper.CreateFileStr

Simulator::Stop(Seconds(20.0));
NS_LOG_INFO("Starting Simulator");
Simulator::Run ();
NS_LOG_INFO("Destroying Simulator");
Simulator::Destroy ();
```



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.1.1	10.1.2.2	TCP	58	49153 → 8080 [SYN] Seq=0 Win=
2	0.008371	10.1.2.2	10.1.1.1	TCP	58	8080 → 49153 [SYN, ACK] Seq=0
3	0.008371	10.1.1.1	10.1.2.2	TCP	54	49153 → 8080 [ACK] Seq=1 Ack=
4	0.008457	10.1.1.1	10.1.2.2	TCP	590	49153 → 8080 [ACK] Seq=1 Ack=
5	0.009401	10.1.1.1	10.1.2.2	TCP	590	49153 → 8080 [ACK] Seq=537 Acl
6	0.010345	10.1.1.1	10.1.2.2	TCP	590	49153 → 8080 [ACK] Seq=1073 A
7	0.011289	10.1.1.1	10.1.2.2	TCP	526	49153 → 8080 [ACK] Seq=1609 A
8	0.016640	10.1.1.1	10.1.2.2	TCP	590	49153 → 8080 [ACK] Seq=2081 A
9	0.017584	10.1.1.1	10.1.2.2	TCP	558	49153 → 8080 [ACK] Seq=2617 A
10	0.018518	10.1.2.2	10.1.1.1	TCP	54	8080 → 49153 [ACK] Seq=1 Ack=
11	0.020406	10.1.2.2	10.1.1.1	TCP	54	8080 → 49153 [ACK] Seq=1 Ack=
12	0.024960	10.1.1.1	10.1.2.2	TCP	590	49153 → 8080 [ACK] Seq=3121 A
13	0.025904	10.1.1.1	10.1.2.2	TCP	558	49153 → 8080 [ACK] Seq=3657 A

```
Header Checksum: 0x0000 [validation disabled]
[Header checksum status: Unverified]
Source Address: 10.1.1.1
Destination Address: 10.1.2.2
```

Transmission Control Protocol, Src Port: 49153, Dst Port: 8080, Seq: 0, Len: 0

```
Source Port: 49153
Destination Port: 8080
[Stream index: 0]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 0
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1001 .... = Header Length: 36 bytes (9)
Flags: 0x002 (SYN)
Window: 65535
[Calculated window size: 65535]
Checksum: 0x0000 [unverified]
[Checksum Status: Unverified]
```

```
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01  .!E..8.. ..@.....
0010 01 01 0a 01 02 02 c0 01 1f 90 00 00 00 00 00 00  .@.....
```

Generating Data and Plotting the Data into Graph

Working with Gnuplot

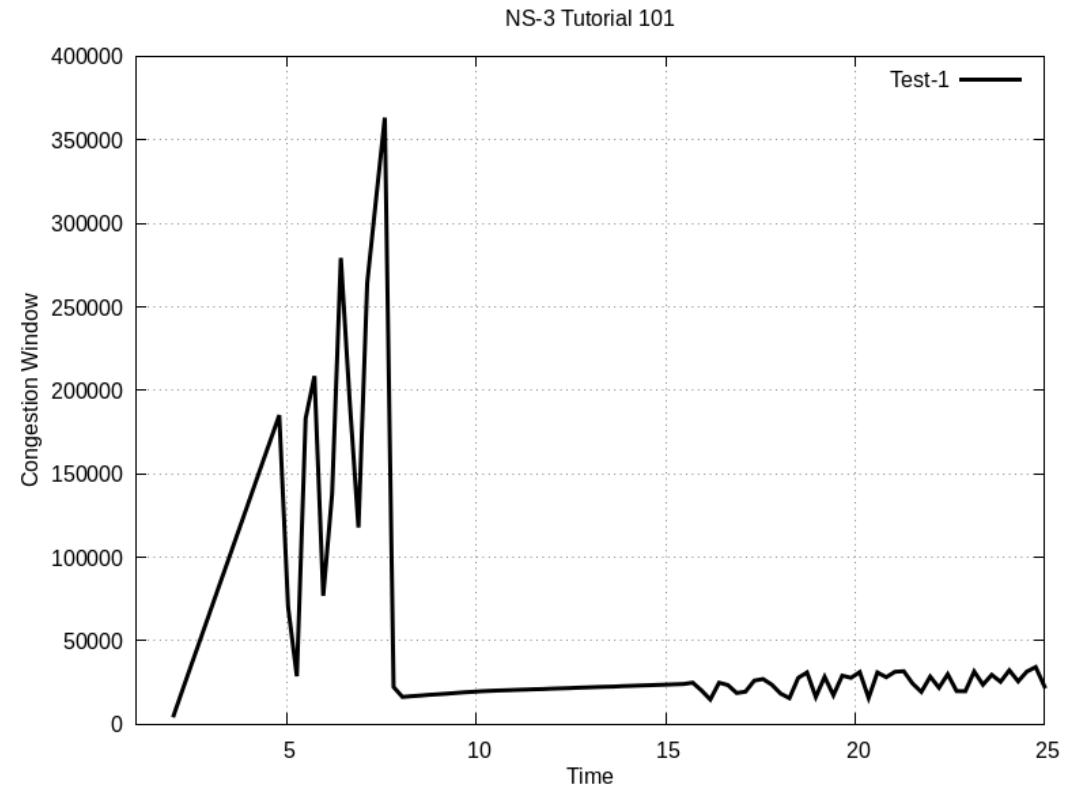
- Generate result.dat file

```
./waf --run scratch/2 >& result.dat
```

- Simple script to Generate Plot

```
gnuplot plot.gnu
```

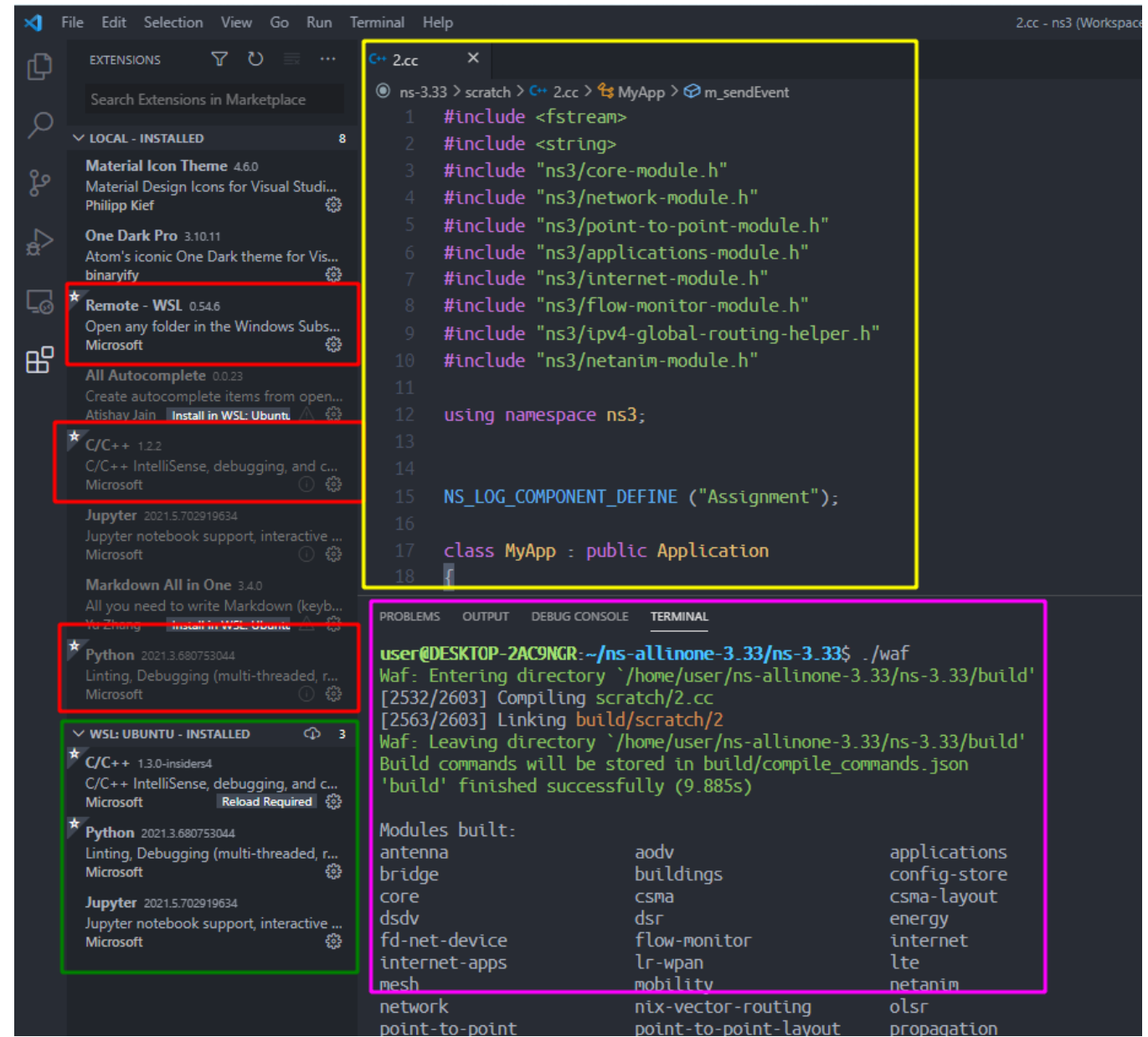
```
1  reset
2  set terminal wxt size 800,600 font 'Liberation Sans,12'
3  set autoscale
4  set key width -2
5  set grid
6  set key opaque right top horizontal
7
8  set xrange [1:25]
9  set title "NS-3 Tutorial 101"
10 set xlabel "Time" offset 0,0.5
11 set ylabel "Congestion Window" offset 1,0
12
13 plot "result.dat" using 1:2 title "Test-1" lc rgb '#4B96D1' lw 3 smooth cspline
14
15 pause -1
```



Use Code Editor based on Your Choice

- There is no specific Code Editor for NS-3.
 - Visual Studio code
 - PyCharm
 - Atom
 - Eclipse

- VSCode has better integration with WSL.



The screenshot shows the Visual Studio Code interface. The left sidebar displays the 'EXTENSIONS' view with a list of installed and available extensions. The main editor area shows a C++ file named '2.cc' with code for an NS-3 application. The bottom panel shows the 'TERMINAL' view with the output of the 'waf' build command.

EXTENSIONS:

- LOCAL - INSTALLED (8)
- Material Icon Theme 4.6.0
- Material Design Icons for Visual Studi... Philipp Kief
- One Dark Pro 3.10.11
- Atom's iconic One Dark theme for Vis... binaryify
- Remote - WSL 0.54.6 (highlighted with a red box)
- All Autocomplete 0.0.23
- C/C++ 1.2.2 (highlighted with a red box)
- Jupyter 2021.5.702919634
- Markdown All in One 3.4.0
- Python 2021.3.680753044 (highlighted with a red box)
- WSL: UBUNTU - INSTALLED (3)
- C/C++ 1.3.0-insiders4
- Python 2021.3.680753044
- Jupyter 2021.5.702919634

Code Editor (2.cc):

```
1 #include <fstream>
2 #include <string>
3 #include "ns3/core-module.h"
4 #include "ns3/network-module.h"
5 #include "ns3/point-to-point-module.h"
6 #include "ns3/applications-module.h"
7 #include "ns3/internet-module.h"
8 #include "ns3/flow-monitor-module.h"
9 #include "ns3/ipv4-global-routing-helper.h"
10 #include "ns3/netanim-module.h"
11
12 using namespace ns3;
13
14 NS_LOG_COMPONENT_DEFINE ("Assignment");
15
16 class MyApp : public Application
17 {
18
```

TERMINAL:

```
user@DESKTOP-2AC9NGR:~/ns-allinone-3.33/ns-3.33$ ./waf
Waf: Entering directory `/home/user/ns-allinone-3.33/ns-3.33/build'
[2532/2603] Compiling scratch/2.cc
[2563/2603] Linking build/scratch/2
Waf: Leaving directory `/home/user/ns-allinone-3.33/ns-3.33/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (9.885s)

Modules built:
antenna          aodv             applications
bridge           buildings        config-store
core             csma             csma-layout
dsv              dsr              energy
fd-net-device    flow-monitor     internet
internet-apps   lr-wpan          lte
mesh            mobility         netanim
network          nix-vector-routing olsr
point-to-point  point-to-point-layout propagation
```

1. **WSL2 GUI X-Server Using VcXsrv** <https://www.shogan.co.uk/how-tos/wsl2-gui-x-server-using-vcxsr/>
2. Windows Subsystem for Linux Installation Guide for Windows 10 <https://docs.microsoft.com/en-us/windows/wsl/install-win10>
3. **WSL-1 and WSL-2 Tutorial** <https://github.com/QMonkey/wsl-tutorial>
4. ns3 Shared Resource by Adil Alsuhaim <https://github.com/addola/NS3-HelperScripts/>
5. Dev on Windows with WSL <https://dowww.spencerwoo.com/>
6. WSL 2 Networking <https://davidbombal.com/wsl-2-networking/>
7. NS3在WSL上的安装 <https://zhuanlan.zhihu.com/p/265510752>
8. NS3 installation https://shihchun.github.io/ns3_installation/
9. NS3 User Groups <https://groups.google.com/g/ns-3-users/>
10. Comparing TCP algorithms <https://haltaro.github.io/comparing-tcp-algorithms/>
11. ns-3 Network Simulator <https://www.youtube.com/watch?v=2W5mdzQrwXI>



北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Thank you for listening!

Contact: mkarim@bit.edu.cn

Feel free to direct your questions
about installation of ns-3 to me

北京理工大学

BEIJING INSTITUTE
OF TECHNOLOGY