LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE
OF COMPUTER APPLICATION, MUMBAI

Network Simulator -3 (NS-3) Practical Lab Manual

VIKRAM PATALBANSI

Faculty, Master of Computer Application (M.C.A.)
Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application

16LEAVES

NETWORK SIMULATOR-3 (NS-3) PRACTICAL LAB MANUAL

NETWORK SIMULATOR-3 (NS-3) PRACTICAL LAB MANUAL

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER APPLICATION, BANDRA, EAST MUMBAI-51

Author: Vikram Patalbansi

Faculty, Master of Computer Application (M.C.A.)
Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application

16LEAVES

First Edition, 2023

Paperback ISBN: 978-81-19221-78-3
eBook ISBN: 978-81-19221-82-0
WebPDF ISBN: 978-81-19221-80-6

Printed and bound in India by
16Leaves
2/579, Singaravelan Street
Chinna Neelankarai
Chennai – 600 041, India

info@16leaves.com
www.16Leaves.com
Call: 91-9940638999

Contents

Practical 1: Installation of Ubuntu and NS3

Networking with Linux

In this we are going to learn network topology logical working using NS-3 and Wireshark simulator.

This simulator nothing but software model in analogus to hardware working of network.

For. Transmission of Packet over Network node shown in NS-3 GUI window.

To download NS-3 refer below website https://www.nsnam.org/

https://www.nsnam.org/

Step 1.

On Ubuntu Terminal type command

sudo apt update

Step 2. After fetching latest package install it command is
sudo apt install

Step 3. After installation upgrade Ubuntu command is
sudo apt upgrade

Step 4. To Install C++ complier in Ubuntu then follows following command
sudo apt-get install build-essential

Step 5: To install Python compiler use following command sudo apt-get install python python-dev

Step 6: To install Mercurial means animation and NS3 supported package use following commands
sudo apt-get install mercurial

Step 7: To unzip file install supported package command is
sudo apt-get install bzr

Step 8: to install TCPDUMP for pcap trace file reading support use following command
sudo apt-get install tcpdump

Step 9: to Generate XML supported file to be readble file external simulator API use the following command
sudo apt-get install libxml2 libxml2-dev

Step 10: Unzip the NS3 package we have command
tar xjf ns-allinone3.32.tar.bz2

Step 11: To compile all the package in ns-3.32 execute following command

./waf configure --enable-examples –enable-tests

Step 12: To run our simulator then every classess must be compile. Use the following command

```
./waf –-run hello-simulator
```

After executing above commands then output will " Hello Simulator" then assume that all NS3 API are in executable code of Operating System.

```
$
$
$
$
$ cd
mkdir workspace cd workspace
wget https://www.nsnam.org/release/ns-allinone-3.32.tar.bz2

tar xjf ns-allinone-3.32.tar.bz2
```

Notice the use above of the wget utility, which is a command-line tool to fetch objects from the web; if you do not have this installed, you can use a browser for this step.
Following these steps, if you change into the directory ns-allinone-3.32, you should see a number of files and directories

```
$ cd ns-allinone-3.32
$ ls

bake constants.py
build.py netanim-3.108 ns-3.32
pybindgen-0.21.0 README
util.py
```

You are now ready to build the base ns-3 distribution and may skip ahead to the section on building ns-3.

Now go ahead and switch back to the debug build that includes the examples and tests.

```
$ ./waf clean
```

to compile all API classes use following commands

```
$ ./waf configure --build-profile=debug --enable-examples –enable-tests
```

if you don't see the "Hello Simulator" output, type the following:

```
$ ./waf configure --build-profile=debug --enable-examples –enable-tests
```

and ten type command

```
./waf configure –enables-examples
```

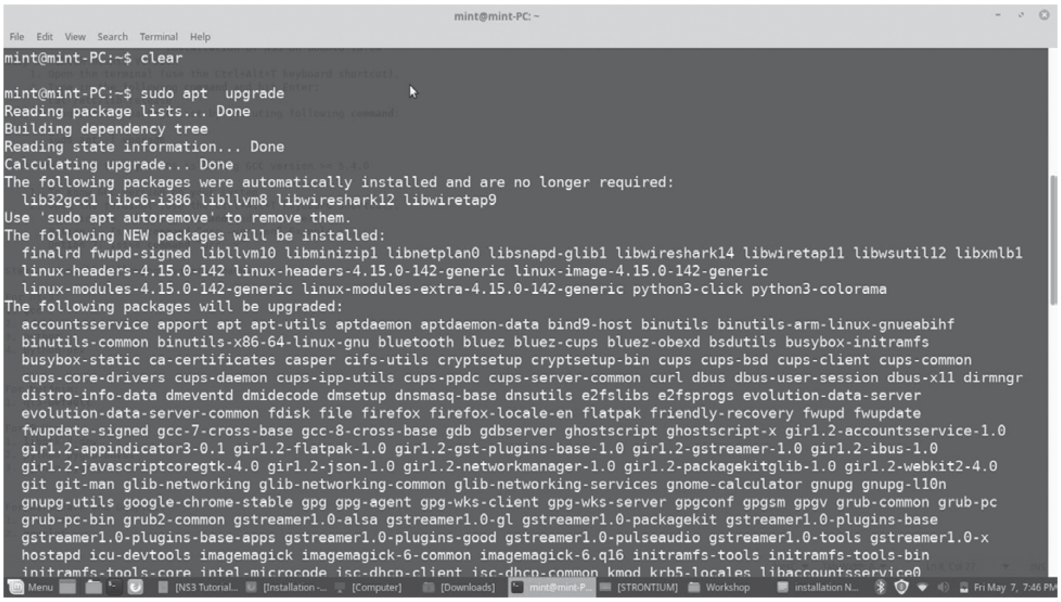
and

```
./waf --run hello-simulator
```

Practical 2: List of Packages for Installing NS-3 in Ubuntu/Mint Systems

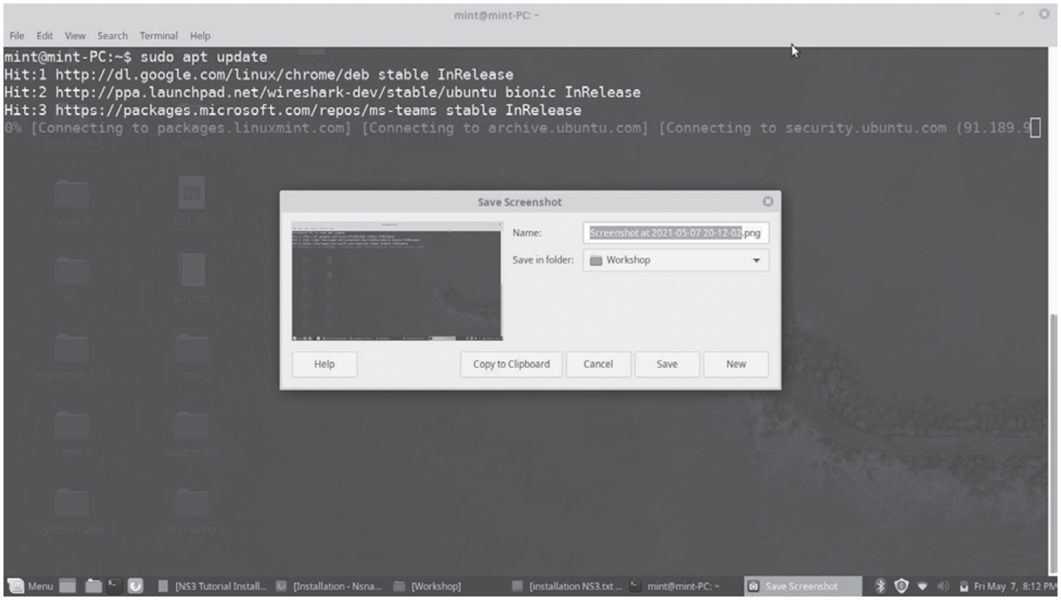
Perquisite for installing NS3.32

1 sudo apt upgrade



```
mint@mint-PC:~$ clear

mint@mint-PC:~$ sudo apt  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:
  lib32gcc1 libc6-i386 libllvm8 libwireshark12 libwiretap9
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  finalrd fwupd-signed libllvm10 libminizip1 libnetplan0 libsnapd-glib1 libwireshark14 libwiretap11 libwsutil12 libxmlb1
  linux-headers-4.15.0-142 linux-headers-4.15.0-142-generic linux-image-4.15.0-142-generic
  linux-modules-4.15.0-142-generic linux-modules-extra-4.15.0-142-generic python3-click python3-colorama
The following packages will be upgraded:
  accountsservice apport apt apt-utils aptdaemon aptdaemon-data bind9-host binutils binutils-arm-linux-gnueabihf
  binutils-common binutils-x86-64-linux-gnu bluetooth bluez bluez-cups bluez-obexd bsdutils busybox-initramfs
  busybox-static ca-certificates casper cifs-utils cryptsetup cryptsetup-bin cups cups-bsd cups-client cups-common
  cups-core-drivers cups-daemon cups-ipp-utils cups-ppdc cups-server-common curl dbus dbus-user-session dbus-x11 dirmngr
  distro-info-data dmeventd dmidecode dmsetup dnsmasq-base dnsutils e2fslibs e2fsprogs evolution-data-server
  evolution-data-server-common fdisk file firefox firefox-locale-en flatpak friendly-recovery fwupd fwupdate
  fwupdate-signed gcc-7-cross-base gcc-8-cross-base gdb gdbserver ghostscript ghostscript-x gir1.2-accountsservice-1.0
  gir1.2-appindicator3-0.1 gir1.2-flatpak-1.0 gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0 gir1.2-ibus-1.0
  gir1.2-javascriptcoregtk-4.0 gir1.2-json-1.0 gir1.2-networkmanager-1.0 gir1.2-packagekitglib-1.0 gir1.2-webkit2-4.0
  git git-man glib-networking glib-networking-common glib-networking-services gnome-calculator gnupg gnupg-l10n
  gnupg-utils google-chrome-stable gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv grub-common grub-pc
  grub-pc-bin grub2-common gstreamer1.0-alsa gstreamer1.0-gl gstreamer1.0-packagekit gstreamer1.0-plugins-base
  gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-good gstreamer1.0-pulseaudio gstreamer1.0-tools gstreamer1.0-x
  hostapd icu-devtools imagemagick imagemagick-6-common imagemagick-6.q16 initramfs-tools initramfs-tools-bin
  initramfs-tools-core intel-microcode isc-dhcp-client isc-dhcp-common kmod krb5-locales libaccountsservice0
```

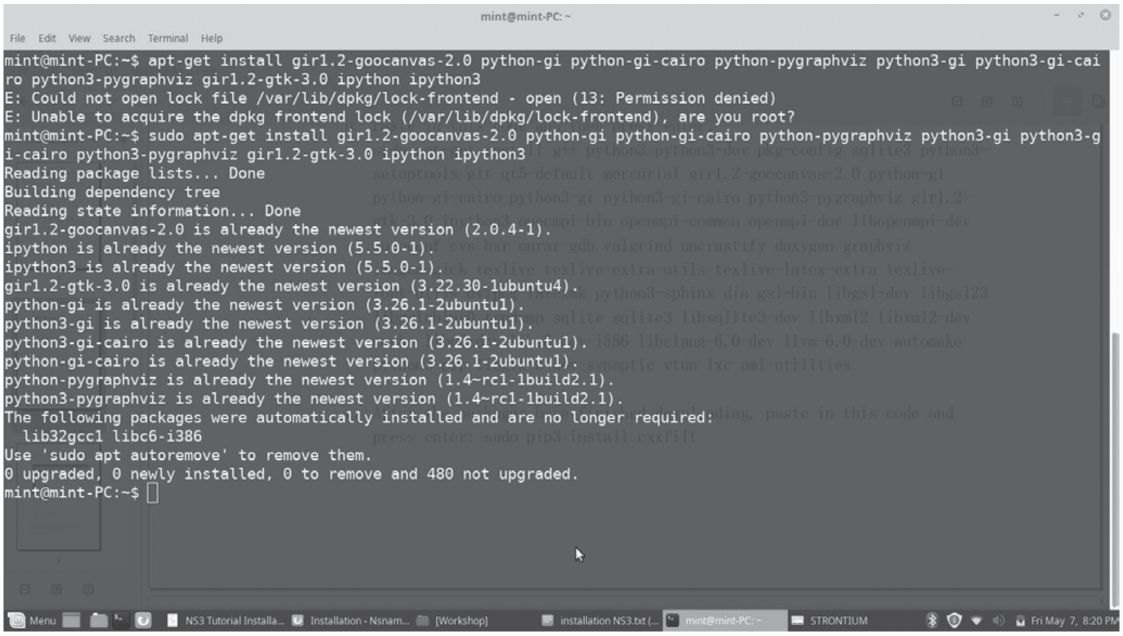
2 Sudo apt update



```
mint@mint-PC:~$ sudo apt update
Hit:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu bionic InRelease
Hit:3 https://packages.microsoft.com/repos/ms-teams stable InRelease
0% [Connecting to packages.linuxmint.com] [Connecting to archive.ubuntu.com] [Connecting to security.ubuntu.com (91.189.9
```

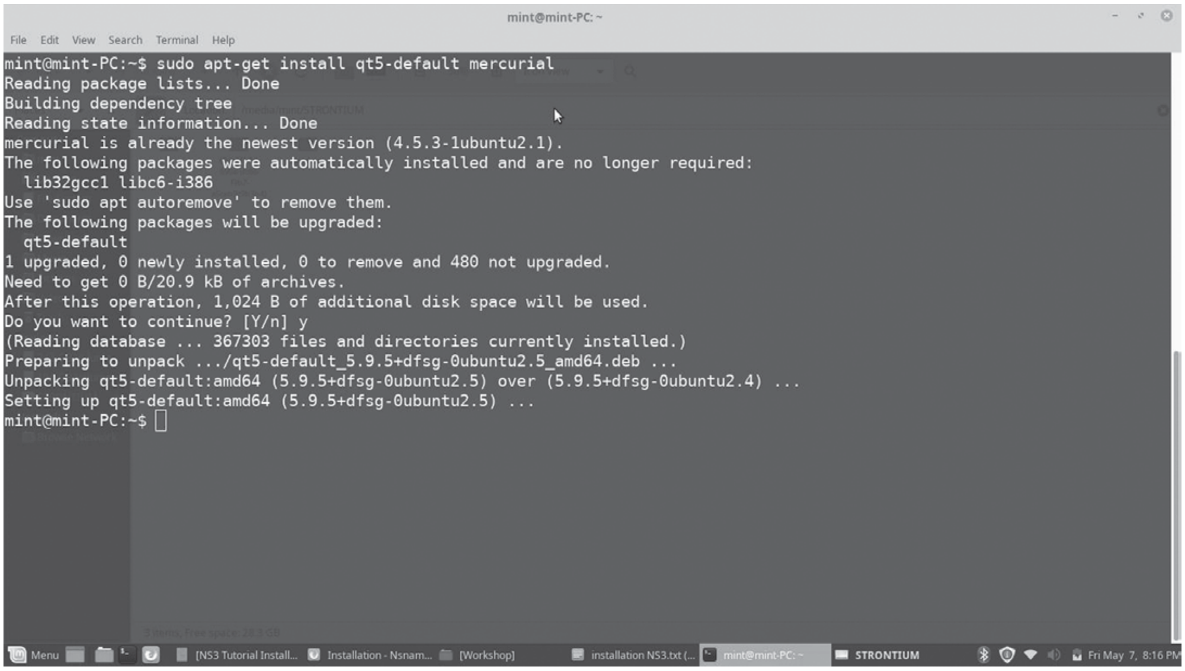
3 Minimal requirements for C++ users
apt-get install g++ python3

4 Minimal requirements for Python API users
apt-get install g++ python3 python3-dev pkg-config sqlite3



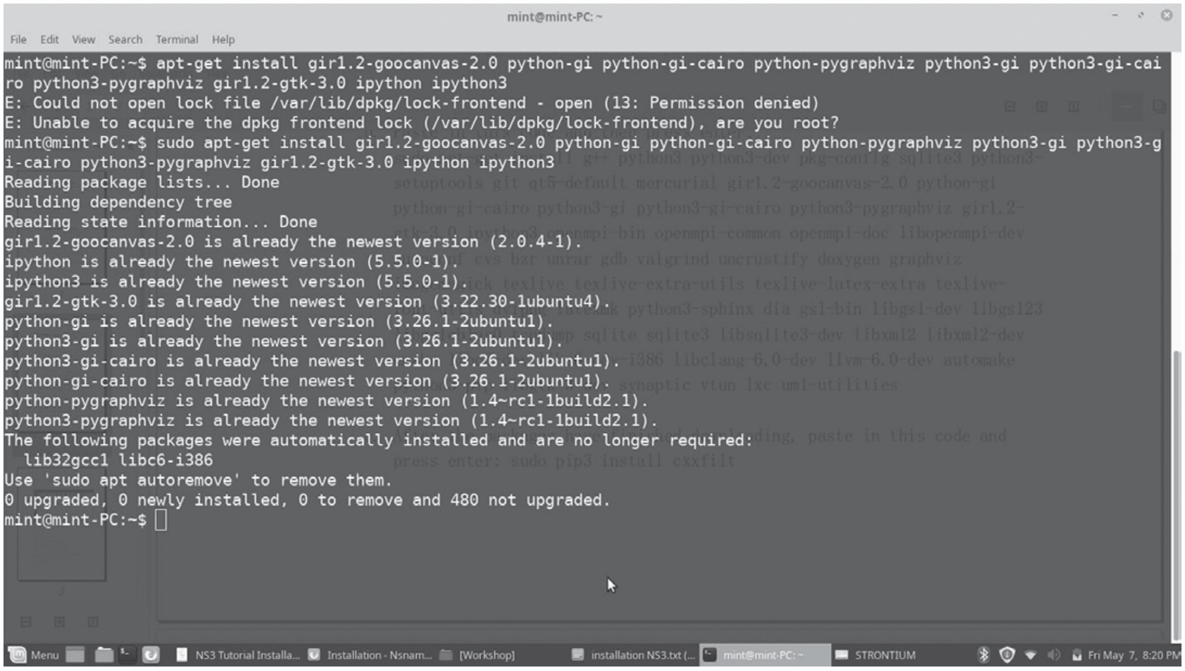
```
mint@mint-PC:~$ apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-gi-cai
ro python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
mint@mint-PC:~$ sudo apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-g
i-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3
Reading package lists... Done
Building dependency tree
Reading state information... Done
gir1.2-goocanvas-2.0 is already the newest version (2.0.4-1).
ipython is already the newest version (5.5.0-1).
ipython3 is already the newest version (5.5.0-1).
gir1.2-gtk-3.0 is already the newest version (3.22.30-1ubuntu4).
python-gi is already the newest version (3.26.1-2ubuntu1).
python3-gi is already the newest version (3.26.1-2ubuntu1).
python3-gi-cairo is already the newest version (3.26.1-2ubuntu1).
python-gi-cairo is already the newest version (3.26.1-2ubuntu1).
python-pygraphviz is already the newest version (1.4~rc1-1build2.1).
python3-pygraphviz is already the newest version (1.4~rc1-1build2.1).
The following packages were automatically installed and are no longer required:
  lib32gcc1 libc6-i386
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 480 not upgraded.
mint@mint-PC:~$
```

5 Netanim animator: qt5 development tools are needed for Netanim animator;
apt-get install qt5-default mercurial



```
mint@mint-PC:~$ sudo apt-get install qt5-default mercurial
Reading package lists... Done
Building dependency tree
Reading state information... Done
mercurial is already the newest version (4.5.3-1ubuntu2.1).
The following packages were automatically installed and are no longer required:
  lib32gcc1 libc6-i386
Use 'sudo apt autoremove' to remove them.
The following packages will be upgraded:
  qt5-default
1 upgraded, 0 newly installed, 0 to remove and 480 not upgraded.
Need to get 0 B/20.9 kB of archives.
After this operation, 1,024 B of additional disk space will be used.
Do you want to continue? [Y/n] y
(Reading database ... 367303 files and directories currently installed.)
Preparing to unpack .../qt5-default_5.9.5+dfsg-0ubuntu2.5_amd64.deb ...
Unpacking qt5-default:amd64 (5.9.5+dfsg-0ubuntu2.5) over (5.9.5+dfsg-0ubuntu2.4) ...
Setting up qt5-default:amd64 (5.9.5+dfsg-0ubuntu2.5) ...
mint@mint-PC:~$
```

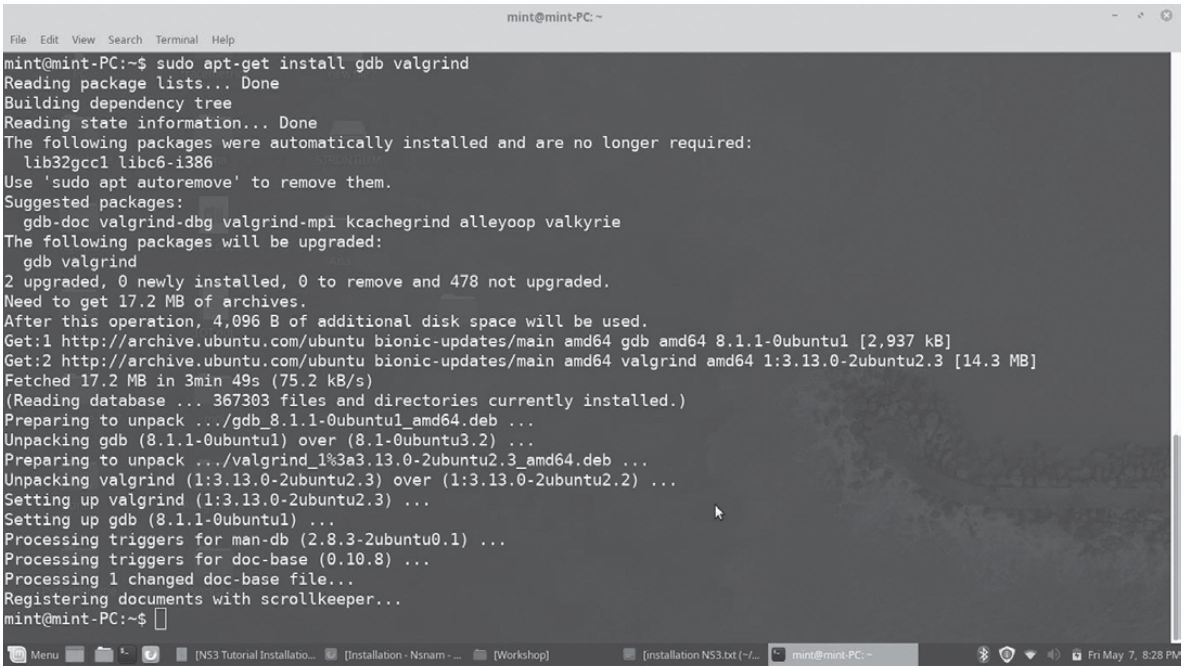
6 ns-3-pyviz visualizer
apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python- pygraphviz python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3



```
mint@mint-PC:~$ apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-gi-cai
ro python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
mint@mint-PC:~$ sudo apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz python3-gi python3-g
i-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython ipython3
Reading package lists... Done
Building dependency tree
Reading state information... Done
gir1.2-goocanvas-2.0 is already the newest version (2.0.4-1).
ipython is already the newest version (5.5.0-1).
ipython3 is already the newest version (5.5.0-1).
gir1.2-gtk-3.0 is already the newest version (3.22.30-1ubuntu4).
python-gi is already the newest version (3.26.1-2ubuntu1).
python3-gi is already the newest version (3.26.1-2ubuntu1).
python3-gi-cairo is already the newest version (3.26.1-2ubuntu1).
python-gi-cairo is already the newest version (3.26.1-2ubuntu1).
python-pygraphviz is already the newest version (1.4~rc1-1build2.1).
python3-pygraphviz is already the newest version (1.4~rc1-1build2.1).
The following packages were automatically installed and are no longer required:
  lib32gcc1 libc6-i386
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 480 not upgraded.
mint@mint-PC:~$ 
```

7 **Debugging:**

8 apt-get install gdb valgrind


mint@mint-PC: ~
File Edit View Search Terminal Help
Menu
[NS3 Tutorial Installatio...
[Installation - Nsnam - ...
[Workshop]
[installation NS3.txt (~/...
mint@mint-PC: ~
Fri May 7, 8:28 PM

```
mint@mint-PC:~$ sudo apt-get install gdb valgrind
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  lib32gcc1 libc6-i386
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  gdb-doc valgrind-dbg valgrind-mpi kcachegrind alleyoop valkyrie
The following packages will be upgraded:
  gdb valgrind
2 upgraded, 0 newly installed, 0 to remove and 478 not upgraded.
Need to get 17.2 MB of archives.
After this operation, 4,096 B of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 gdb amd64 8.1.1-0ubuntu1 [2,937 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 valgrind amd64 1:3.13.0-2ubuntu2.3 [14.3 MB]
Fetched 17.2 MB in 3min 49s (75.2 kB/s)
(Reading database ... 367303 files and directories currently installed.)
Preparing to unpack .../gdb_8.1.1-0ubuntu1_amd64.deb ...
Unpacking gdb (8.1.1-0ubuntu1) over (8.1-0ubuntu3.2) ...
Preparing to unpack .../valgrind_1%3a3.13.0-2ubuntu2.3_amd64.deb ...
Unpacking valgrind (1:3.13.0-2ubuntu2.3) over (1:3.13.0-2ubuntu2.2) ...
Setting up valgrind (1:3.13.0-2ubuntu2.3) ...
Setting up gdb (8.1.1-0ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for doc-base (0.10.8) ...
Processing 1 changed doc-base file...
Registering documents with scrollkeeper...
mint@mint-PC:~$ 
```

9 Doxygen and related inline documentation:

apt-get install doxygen graphviz imagemagick
apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils dvipng latexmk

10 The ns-3 manual and tutorial are written in reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures typically in dia (also needs the texlive packages above):

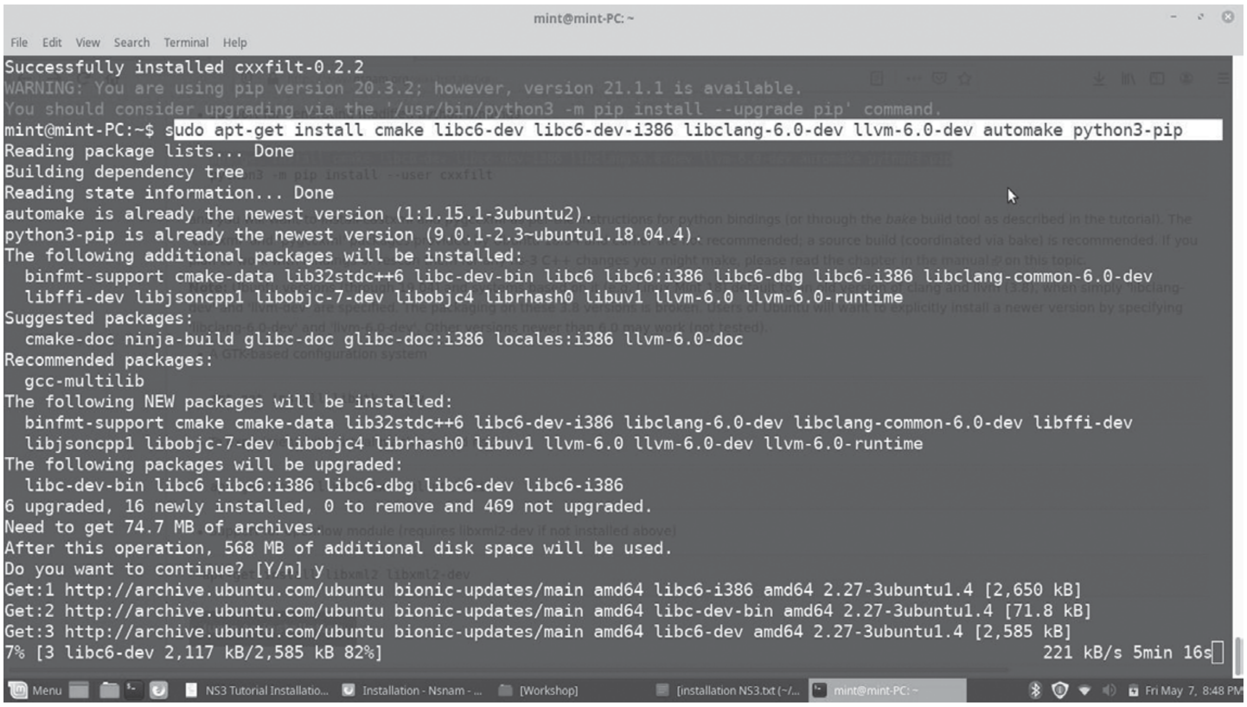
apt-get install python3-sphinx dia

11 To read pcap packet traces

apt-get install tcpdump

12 Support for generating modified python bindings

apt-get install cmake libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake python3-pip
python3 -m pip install -- user cxxfilt


mint@mint-PC: ~
File Edit View Search Terminal Help
Menu
NS3 Tutorial Installatio...
Installation - Nsnam - ...
[Workshop]
[installation NS3.txt (~/...
mint@mint-PC: ~
Fri May 7, 8:48 PM

```
Successfully installed cxxfilt-0.2.2
WARNING: You are using pip version 20.3.2; however, version 21.1.1 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
mint@mint-PC:~$ sudo apt-get install cmake libc6-dev libc6-dev-i386 libclang-6.0-dev llvm-6.0-dev automake python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
automake is already the newest version (1:1.15.1-3ubuntu2).
python3-pip is already the newest version (9.0.1-2.3~ubuntu1.18.04.4).
The following additional packages will be installed:
  binfmt-support cmake-data lib32stdc++6 libc-dev-bin libc6 libc6:i386 libc6-dbg libc6-i386 libclang-common-6.0-dev
  libffi-dev libjsoncpp1 libobjc-7-dev libobjc4 librhash0 libuv1 llvm-6.0 llvm-6.0-runtime
Suggested packages:
  cmake-doc ninja-build glibc-doc glibc-doc:i386 locales:i386 llvm-6.0-doc
Recommended packages:
  gcc-multilib
The following NEW packages will be installed:
  binfmt-support cmake cmake-data lib32stdc++6 libc6-dev-i386 libclang-6.0-dev libclang-common-6.0-dev libffi-dev
  libjsoncpp1 libobjc-7-dev libobjc4 librhash0 libuv1 llvm-6.0 llvm-6.0-dev llvm-6.0-runtime
The following packages will be upgraded:
  libc-dev-bin libc6 libc6:i386 libc6-dbg libc6-dev libc6-i386
6 upgraded, 16 newly installed, 0 to remove and 469 not upgraded.
Need to get 74.7 MB of archives.
After this operation, 568 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc6-i386 amd64 2.27-3ubuntu1.4 [2,650 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc-dev-bin amd64 2.27-3ubuntu1.4 [71.8 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libc6-dev amd64 2.27-3ubuntu1.4 [2,585 kB]
7% [3 libc6-dev 2,117 kB/2,585 kB 82%]                                                        221 kB/s 5min 16s
```

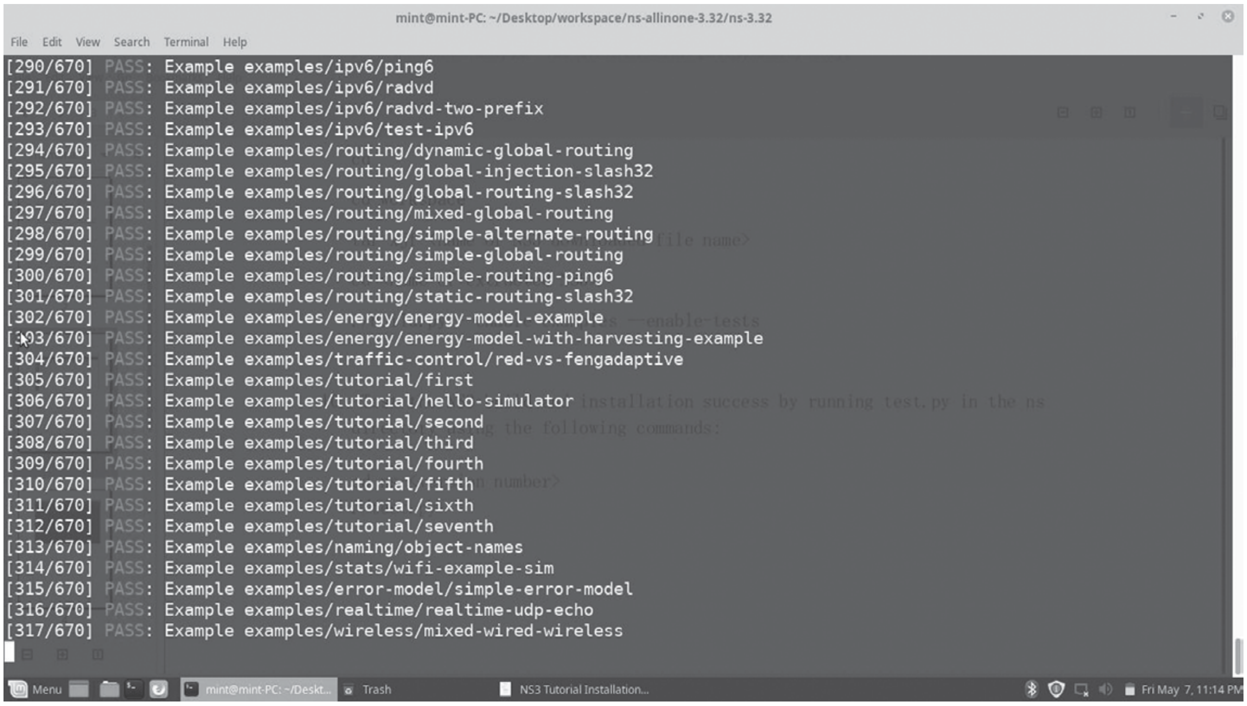
After installing the required packages, create a folder named **workspace** in the home directory and then put the NS3 tar package into the workspace.
Go to terminal and input these commands consecutively after each command finishes executing:

```
cd
cd workspace
tar xjf <name of NS3 downloaded file name>

cd <name of extracted NS3>

./build.py --enable-examples --enable-tests
```

It takes time be patient!!

Test the NS3 build and installation success by running test.py in the ns directory using the following commands:

cd ns-<version number>./test.py


mint@mint-PC: ~/Desktop/workspace/ns-allinone-3.32/ns-3.32
File Edit View Search Terminal Help
Menu
mint@mint-PC: ~/Deskt...
Trash
NS3 Tutorial Installation...
Fri May 7, 11:14 PM

```
[290/670] PASS: Example examples/ipv6/ping6
[291/670] PASS: Example examples/ipv6/radvd
[292/670] PASS: Example examples/ipv6/radvd-two-prefix
[293/670] PASS: Example examples/ipv6/test-ipv6
[294/670] PASS: Example examples/routing/dynamic-global-routing
[295/670] PASS: Example examples/routing/global-injection-slash32
[296/670] PASS: Example examples/routing/global-routing-slash32
[297/670] PASS: Example examples/routing/mixed-global-routing
[298/670] PASS: Example examples/routing/simple-alternate-routing
[299/670] PASS: Example examples/routing/simple-global-routing
[300/670] PASS: Example examples/routing/simple-routing-ping6
[301/670] PASS: Example examples/routing/static-routing-slash32
[302/670] PASS: Example examples/energy/energy-model-example
[303/670] PASS: Example examples/energy/energy-model-with-harvesting-example
[304/670] PASS: Example examples/traffic-control/red-vs-fengadaptive
[305/670] PASS: Example examples/tutorial/first
[306/670] PASS: Example examples/tutorial/hello-simulator
[307/670] PASS: Example examples/tutorial/second
[308/670] PASS: Example examples/tutorial/third
[309/670] PASS: Example examples/tutorial/fourth
[310/670] PASS: Example examples/tutorial/fifth
[311/670] PASS: Example examples/tutorial/sixth
[312/670] PASS: Example examples/tutorial/seventh
[313/670] PASS: Example examples/naming/object-names
[314/670] PASS: Example examples/stats/wifi-example-sim
[315/670] PASS: Example examples/error-model/simple-error-model
[316/670] PASS: Example examples/realtime/realtime-udp-echo
[317/670] PASS: Example examples/wireless/mixed-wired-wireless
```

Practical 3: Steps for Installing NetAnim and Wireshark Simulator

Installation of NS3 on Ubuntu 18.04/Linux Mint 19

Step 1: Check Ubuntu Version

1. Open the terminal (use the Ctrl+Alt+T keyboard shortcut).
2. Type in the following command and hit Enter: cat /etc/lsb-release

Step 2: Upgrade the package list by executing following command: sudo apt -f upgrade

Note: Make sure that your OS is having GCC version >= 5.4.0

1. To know GCC version on your system
2. Open the terminal (use the Ctrl+Alt+T keyboard shortcut). Type in the following command and hit Enter:
   Option 1. Issue command "gcc –version" Example: ...
   Option 2. Issue command "gcc -v"

Step 3: Install following required packages:

For ns-3:
gcc
g++
python
python-dev

For NetAnim:
qt5-default

For PyViz:
libgtk-3-dev
python-pygoocanvas
python-pygraphviz

For Wireshark and Gnuplot:

wireshark
gnuplot
Command to install all these packages together:

sudo apt-get install gcc g++ python python-dev qt4-dev-tools libgtk-3-dev python- pygoocanvas python-pygraphviz wireshark gnuplot

Step 4: Download ns-allinone-3.27.tar.bz2 and unzip it. For unzip write following command.

tar -xvjf ns-allinone-3.27.tar.bz2

Step 5: Go to ns-allinone-3.27 folder and install ns3: Follow the bellow command to do so:

```
cd ns-allinone-3.27
./build.py --enable-examples –enable-tests
```

This command will install ns-3, NetAnim and PyViz. You will be having a list of built and unbuilt modules after successful installation.

Step 6: Once the installation completes, go to ns-3.27 and give the following command for testing the installation:

```
cd ns-allinone-3.27/ns-3.27/./test.py -c core
./waf –run hello-simulator You are done!
```

Step 7: To install the Wireshark

```
ns-allinone-3.32$sudo apt-get install wireshark
```

Step 8: To install GNUplot

```
ns-allinone-3.32$ sudo apt-get gnuplot
```

Note:
Minimal requirements for C++ users sudo apt-get install g++ python3

Minimal requirements for Python API users

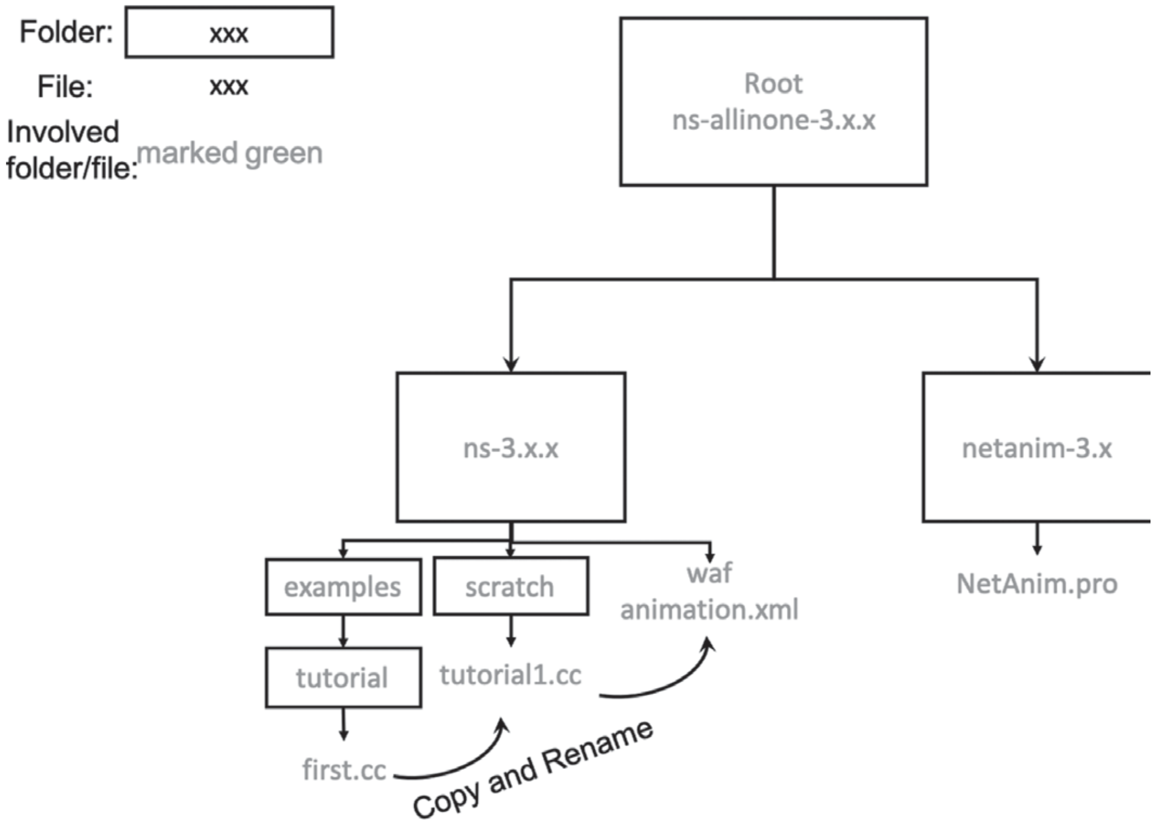
```
sudo apt-get install g++ python3 python3-dev pkg-config sqlite3
```

NetAnim animator: - qt5 development tools are needed for NetAnim animator sudo apt-get install qt5-default mercurial

Practical 4: Configure NetAnim Simulator

Prerequests: **NS3 Installation on Ubuntu 1804**

Related folders and files in this document



Configure NetAnim

```
# Go to NS3 All-in-one folder and go further into NetAnim folder
cd ns-allinone-3.30.1/netanim-3.108/

#
sudo apt-get install make -y

# clean the source code, qmake and then make
make clean
qmake NetAnim.pro make
```

Now, NetAnim is ready to use.

Run Tutorial Examples

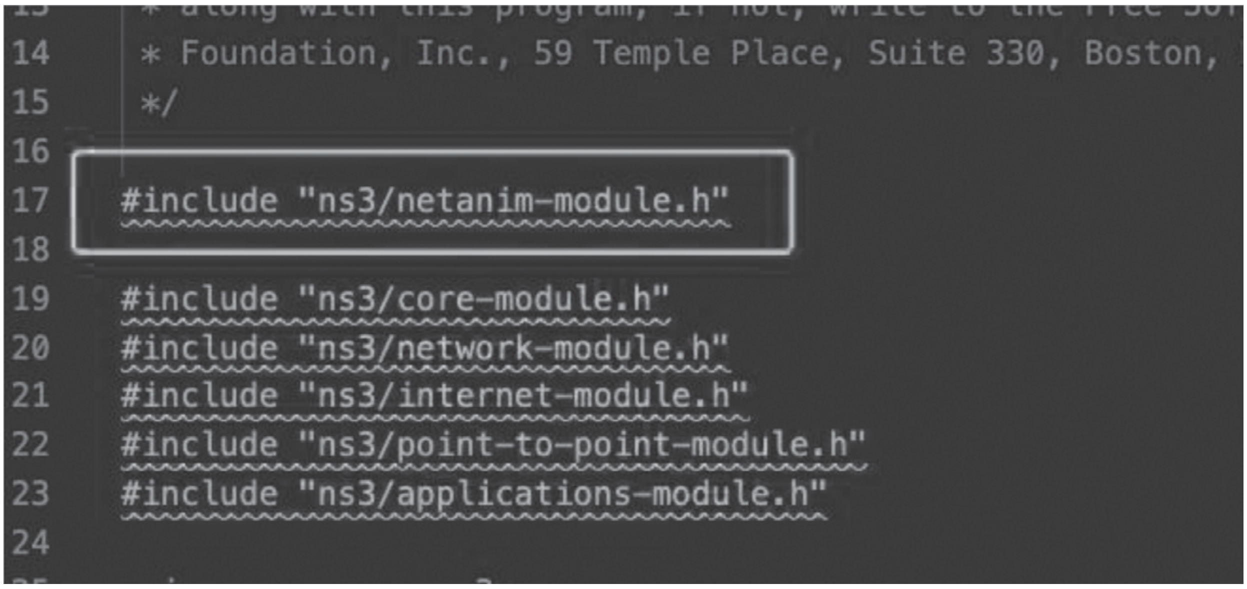
Pay attention to the name of the file

```
# go to ns3-30.1 examples
cd .. # now we are at ns-allinone-3.30.1/
cd ns-3.30.1/examples/tutorial

# instead of ruining the example, we copy it to scratch/ Note that, you cannot change the
folder unless you have configured the path
# Also, you should not reuse the name "first", because NS3 will find out first.cc or first.py
instead of your file.
# here we use tutorial1.cc
cp first.cc ../../scratch/tutorial1.cc cd ../../scratch/

# use your favourite code editor to edit the first.cc file.
# I use vscode.
```

Add Code for NetAnim

1 Add the header file

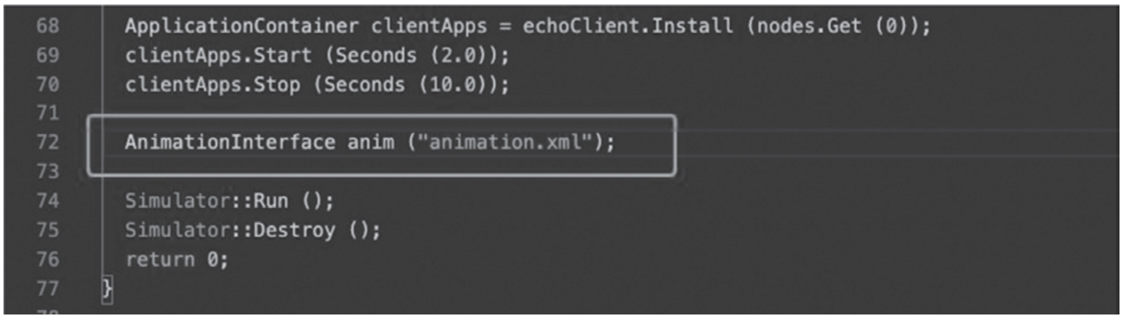
#include “ns3/netanim-module.h”



```
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
 */

#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"
```

2 Add the following statement before Simulation::Run()

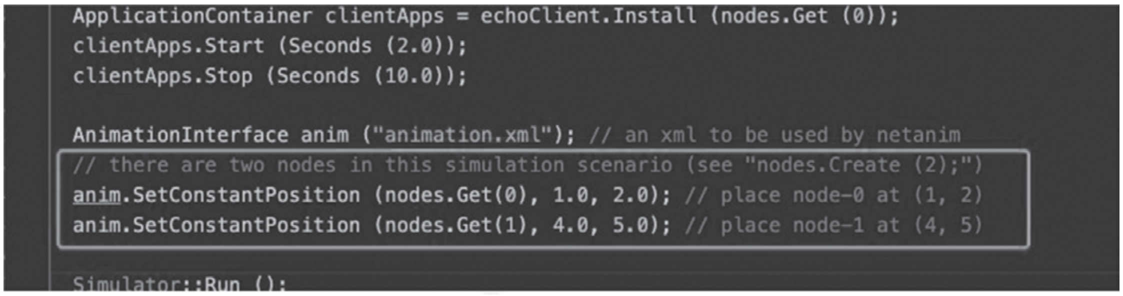
AnimationInterface anim (“animation.xml”);



```
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

AnimationInterface anim ("animation.xml");

Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

3 Set give positions to your nodes.

anim.SetConstantPosition (node, double x, double y);

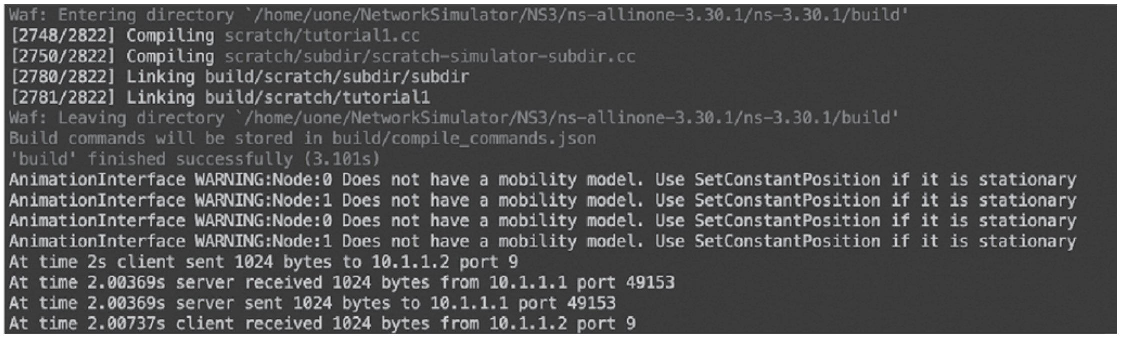


```
ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

AnimationInterface anim ("animation.xml"); // an xml to be used by netanim
// there are two nodes in this simulation scenario (see "nodes.Create (2);")
anim.SetConstantPosition (nodes.Get(0), 1.0, 2.0); // place node-0 at (1, 2)
anim.SetConstantPosition (nodes.Get(1), 4.0, 5.0); // place node-1 at (4, 5)

Simulator::Run ();
```

2.2.3.6. Build and Run

```
# go back to ns-3.30.1
cd ../

# use waf to compile and run the code
./waf --run tutorial1
```

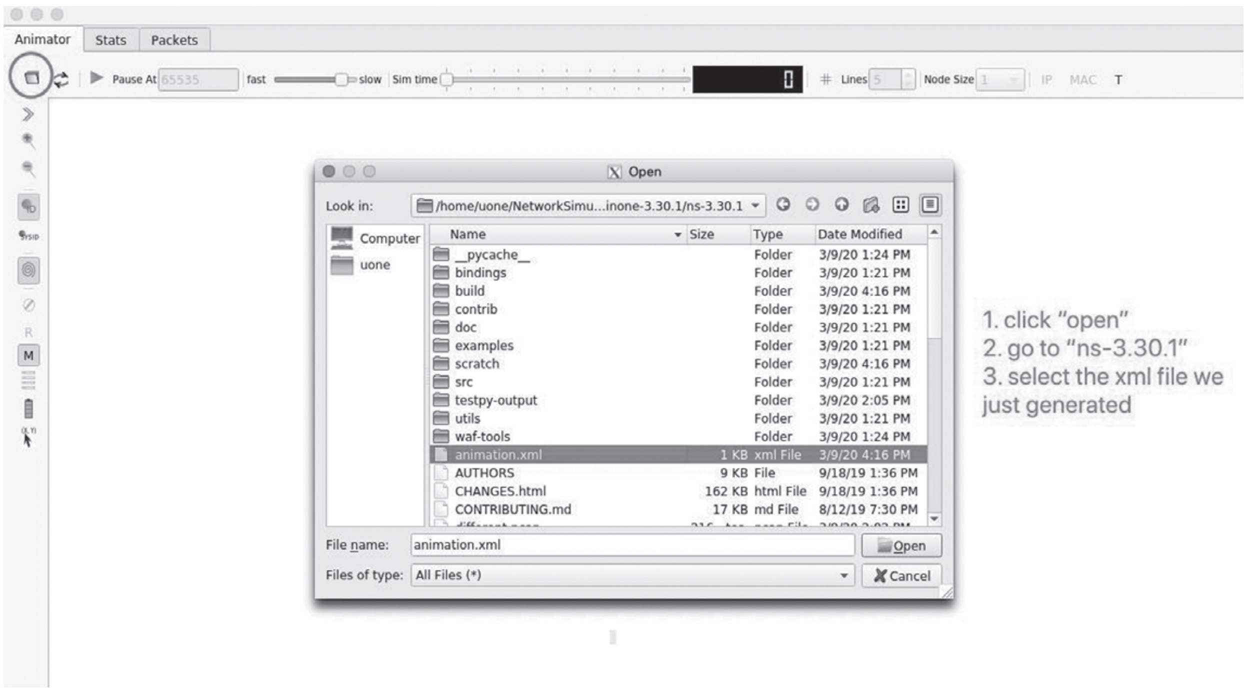
You should see



```
Waf: Entering directory `/home/uone/NetworkSimulator/NS3/ns-allinone-3.30.1/ns-3.30.1/build'
[2748/2822] Compiling scratch/tutorial1.cc
[2750/2822] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2780/2822] Linking build/scratch/subdir/subdir
[2781/2822] Linking build/scratch/tutorial1
Waf: Leaving directory `/home/uone/NetworkSimulator/NS3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.101s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
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
```

Use NetAnim to Show The Animation

```
# go to netanim-3.108
cd ../netanim-3.108/

# run NetAnim
./NetAnim
```



Animator | Stats | Packets

Pause At 65535 | fast | slow | Sim time | Lines 5 | Node Size 1 | IP | MAC | T

Open

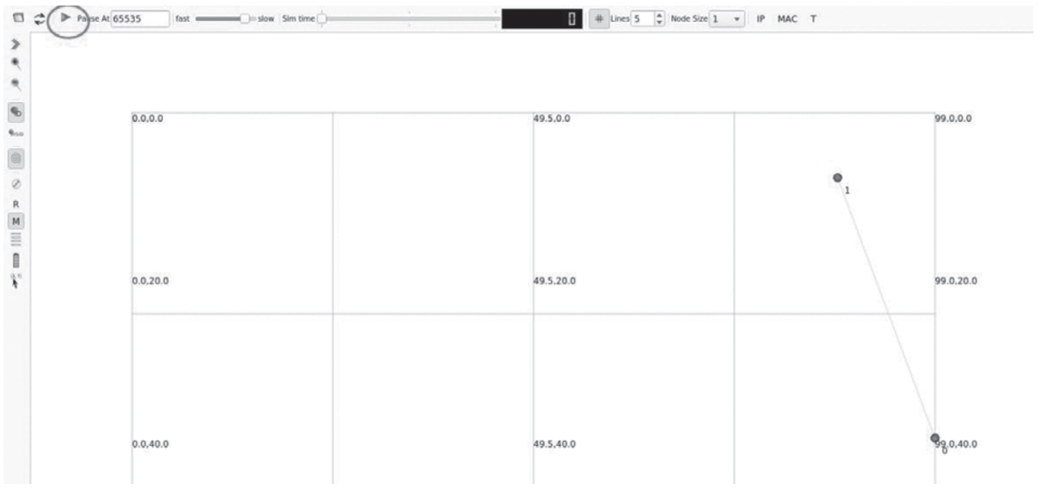
Look in: /home/uone/NetworkSimu...inone-3.30.1/ns-3.30.1

Computer
uone

Name	Size	Type	Date Modified
__pycache__		Folder	3/9/20 1:24 PM
bindings		Folder	3/9/20 1:21 PM
build		Folder	3/9/20 4:16 PM
contrib		Folder	3/9/20 1:21 PM
doc		Folder	3/9/20 1:21 PM
examples		Folder	3/9/20 1:21 PM
scratch		Folder	3/9/20 4:16 PM
src		Folder	3/9/20 1:21 PM
testpy-output		Folder	3/9/20 2:05 PM
utils		Folder	3/9/20 1:21 PM
waf-tools		Folder	3/9/20 1:24 PM
animation.xml	1 KB	xml File	3/9/20 4:16 PM
AUTHORS	9 KB	File	9/18/19 1:36 PM
CHANGES.html	162 KB	html File	9/18/19 1:36 PM
CONTRIBUTING.md	17 KB	md File	8/12/19 7:30 PM

File name: animation.xml
Files of type: All Files (*)
Open
Cancel

1. click "open"
2. go to "ns-3.30.1"
3. select the xml file we just generated



Practical 5: Conceptual Explanations of NS3 code

Conceptual Overview

Node

In Internet jargon, a computing device that connects to a network is called a host or sometimes an end system. Because |ns3| is a network simulator, not specifically an Internet simulator, we intentionally do not use the term host since it is closely associated with the Internet and its protocols.
Instead, we use a more generic term also used by other simulators that originates in Graph Theory --- the node.

In |ns3| the basic computing device abstraction is called the node. This abstraction is represented in C++ by the class Node.

The Node class provides methods for managing the representations of computing devices in simulations.
You should think of a Node as a computer to which you will add functionality. One adds things like applications, protocol stacks and peripheral cards with their associated drivers to enable the computer to do useful work. We use the same basic model in |ns3|.

Application

Typically, computer software is divided into two broad classes. System Software organizes various computer resources such as memory, processor cycles, disk, network, etc., according to some computing model. System software usually does not use those resources to complete tasks that directly benefit a user. A user would typically run an application that acquires and uses the resources controlled by the system software to accomplish some goal. Often, the line of separation between system and application software is made at the privilege level change that happens in operating system traps. In |ns3| there is no real concept of operating system and especially no concept of privilege levels or system calls. We do, however, have the idea of an application.
Just as software applications run on computers to perform tasks in the "real world," |ns3| applications run on |ns3|

Nodes to drive simulations in the simulated world.

In |ns3| the basic abstraction for a user program that generates some activity to be simulated is the application.

This abstraction is represented in C++ by the class Application.
The Application class provides methods for managing the representations of our version of user-level applications in simulations.

Developers are expected to specialize the Application class in the object-oriented programming sense to create new applications.

In this tutorial, we will use specializations of class Application called UdpEchoClientApplication and UdpEchoServerApplication.

As you might expect, these applications compose a client/server application set used to generate and echo simulated network packets

Channel

In the real world, one can connect a computer to a network. Often the media over which data flows in these networks are called channels. When you connect your Ethernet cable to the plug in the wall, you are connecting your computer to an Ethernet communication channel. In the simulated world of |ns3|, one connects a Node to an object representing a communication channel.

Here the basic communication subnetwork abstraction is called the channel and is represented in C++ by the class Channel.

The Channel class provides methods for managing communication subnetwork objects and connecting nodes to them. Channels may also be specialized by developers in the object oriented programming sense.

A Channel specializationmay model something as simple as a wire. The specialized Channel can also model things as complicated as a large Ethernet switch, or three-dimensional space full of obstructions in the case of wireless networks.

We will use specialized versions of the Channel called CsmaChannel, PointToPointChannel and WifiChannel.

The CsmaChannel, for example, models a version of a communication subnetwork that implements a carrier sense multiple access communication medium. This gives us Ethernet-like functionality.

Net Device

It used to be the case that if you wanted to connect a computer to a network, you had to buy a specific kind of network cable and a hardware device called (in PC terminology) a peripheral card that needed to be installed in your computer. If the peripheral card implemented some networking function, they were called Network Interface Cards, or NICs.

Today mostcomputers come with the network interface hardware built in and users don't see these building blocks.

A NIC will not work without a software driver to control the hardware. In Unix (or Linux), a piece of peripheral hardware is classified as a device. Devices are controlled using device drivers, and network devices (NICs) are controlled

using network device drivers collectively known as net devices.

In Unix and Linux you refer to these net devices by names such as eth0.

In |ns3| the net device abstraction covers both the software driver and the simulated hardware. A net device is "installed" in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels.

Just as in a real computer, a Node may be connected to more than one Channel via multiple NetDevices.

The net device abstraction is represented in C++ by the class NetDevice.

The NetDevice class provides methods for managing connections to Node and Channel objects; and may be specialized by developers in the object-oriented programming sense. We will use the several specialized versions of the NetDevice called CsmaNetDevice, PointToPointNetDevice, and WifiNetDevice.

Just as an Ethernet NIC is designed to work with an Ethernet network, the CsmaNetDevice is designed to work with a CsmaChannel;
the PointToPointNetDevice is designed to work with a PointToPointChannel and a WifiNetNevice is designed to work with a WifiChannel.

Topology Helpers

In a real network, you will find host computers with added (or built-in) NICs.

In |ns3| we would say that you will find Nodes with attached NetDevices. In a large simulated network you will need to arrange many connections between Nodes, NetDevices and Channels.

Since connecting NetDevices to Nodes, NetDevices to Channels, assigning IP addresses, etc., are such common tasks in |ns3|, we provide what we call topology helpers to make this as easy as possible.

For example, it may take many distinct |ns3| core operations to create a NetDevice, add a MAC address, install that net device on a Node, configure the node's protocol stack, and then connect the NetDevice to a Channel. Even more operations would be required to connect multiple devices onto multipoint channels and then to connect individual networks together into internetworks.

We provide topology helper objects that combine those many distinct operations into an easy to use model for your convenience.

A First ns-3 Script

4If you downloaded the system as was suggested above, you will have a release of
|ns3| in a directory called repos under your home directory. Change into that release directory, and you should find a directory structure something like the following:

AUTHORS examples scratch utils waf.bat* bindings LICENSE src utils.py waf-tools build ns3
test.py* utils.pyc wscript
CHANGES.html README
testpy-output VERSION wutils.py
doc
RELEASE_NOTES testpy.supp waf*
wutils.pyc

Change into the examples/tutorial directory. You should see a file named first.cc located there. This is a script that will create a simple point-to-point link between two nodes and echo a single packet between the nodes. Let's take a look at that script line by line, so go ahead and open first.cc in your favourite editor.

Boilerplate

The first line in the file is an emacs mode line. This tells emacs about the formatting conventions (coding style) we use in our source code.

/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */

This is always a somewhat controversial subject, so we might as well get it out of the way immediately. The |ns3| project, like most large projects, has adopted a coding style to which all contributed code must adhere. If you want to contribute your code to the project, you will eventually have to conform to the |ns3| coding standard as described in the file doc/codingstd.txt or shown on the project web page here.

We recommend that you, well, just get used to the look and feel of |ns3| code and adopt this standard whenever you are working with our code. All of the development team and contributors have done so with various amounts of grumbling. The emacs mode line above makes it easier to get the formatting correct if you use the emacs editor.

The |ns3| simulator is licensed using the GNU General Public License. You will see the appropriate GNU legalese at the head of every file in the |ns3| distribution.

Often you will see a copyright notice for one of the institutions involved in the |ns3| project above the GPL text and an author listed below.

```
/*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/
```

Module Includes

The code proper starts with a number of include statements.

```
#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"
```

To help our high-level script users deal with the large number of include files present in the system, we group includes according to relatively large modules. We provide a single include file that will recursively load all of the include files used
in each module. Rather than having to look up exactly what header you need, and possibly have to get a number of dependencies right, we give you the ability to load a group of files at a large granularity.
This is not the most efficient approach but it certainly makes writing scripts much easier.
Each of the |ns3| include files is placed in a directory called ns3 (under the build directory) during the build process to help avoid include file name collisions.

The ns3/core-module.h file corresponds to the ns-3 module you will find in the directory src/core in your downloaded release distribution.
If you list this directory you will find a large number of header files.

When you do a build, Waf will place public header files in an ns3 directory under the appropriate build/debug or build/optimized directory depending on your configuration. Waf will also automatically generate a module include file to load all of the public header files.

Since you are, of course, following this tutorial religiously, you will already have done a

```
$ ./waf -d debug --enable-examples --enable-tests configure
```

in order to configure the project to perform debug builds that include examples and tests. You will also have done a

```
$ ./waf
```

to build the project.

So now if you look in the directory ../../build/debug/ns3

you will find the four module include files shown above. You can take a look at the contents of these files and find that they do include all of the public include files in their respective modules.

Ns3 Namespace

The next line in the first.cc script is a namespace declaration.

```
using namespace ns3;
```

The |ns3| project is implemented in a C++ namespace called ns3.
This groups all |ns3|-related declarations in a scope outside the global namespace, which we hope will help with integration with other code.
The C++ using statement introduces the |ns3| namespace into the current (global) declarative region.
This is a fancy way of saying that after this declaration, you will not have to type ns3:: scope resolution operator before all of the |ns3| code in order to use it.

If you are unfamiliar with namespaces, please consult almost any C++ tutorial and compare the ns3 namespace and usage here with instances of the std namespace
and the using namespace std; statements you will often find in discussions of cout and streams.

Logging

The next line of the script is the following,

```
NS_LOG_COMPONENT_DEFINE (“FirstScriptExample”);
```

We will use this statement as a convenient place to talk about our Doxygen documentation system.
If you look at the project web site, ns-3 project, you will find a link to “Documentation” in the navigation bar. If you select this link, you will be taken to our documentation page.

There is a link to “Latest Release” that will take you to the documentation for the latest stable release of |ns3|.
If you select the “API Documentation” link, you will be taken to the |ns3| API documentation page.
Along the left side, you will find a graphical representation of the structure of the documentation.
A good place to start is the NS-3 Modules “book” in the |ns3| navigation tree.

If you expand Modules you will see a list of |ns3| module documentation. The concept of module here ties directly into the module include files discussed above.

The |ns3| logging subsystem is discussed in the:ref:`UsingLogging` section, so we'll get to it later in this tutorial, but you can find out about the above statement by looking at the Core module, then expanding the Debugging tools book, and then selecting the Logging page.

Click on Logging.

You should now be looking at the Doxygen documentation for the Logging module.
In the list of Macros's at the top of the page you will see the entry for NS_LOG_COMPONENT_DEFINE.

Before jumping in, it would probably be good to look for the "Detailed Description" of the logging module to get a feel for the overall operation. You can either scroll down or select the "More..." link under the collaboration diagram to do this.

Once you have a general idea of what is going on, go ahead and
take a look at the specific NS_LOG_COMPONENT_DEFINE documentation.

It summarize, this line declares a logging component called FirstScriptExample that allows you to enable and disable console message logging by reference to the name.

Main Function

The next lines of the script you will find are,

```
int
main (int argc, char *argv[])
{
```

This is just the declaration of the main function of your program (script). Just as in any C++ program, you need to define a main function that will be the first function run. There is nothing at all special here. Your |ns3| script is just a C++ program.
The next line sets the time resolution to one nanosecond, which happens to be the default value:

```
Time::SetResolution (Time::NS);
```

The resolution is the smallest time value that can be represented (as well as the smallest representable difference between two time values).

You can change the resolution exactly once. The mechanism enabling this flexibility is somewhat memory hungry, so once the resolution has been set explicitly we release the memory, preventing further updates.

(If you don't set the resolution explicitly, it will default to one nanosecond, and the memory will be released when the simulation starts.)

The next two lines of the script are used to enable two logging components that are built into the Echo Client and Echo Server applications:

```
LogComponentEnable("UdpEchoClientApplication",
LOG_LEVEL_INFO);

LogComponentEnable("UdpEchoServerApplication",
LOG_LEVEL_INFO);
```

If you have read over the Logging component documentation you will have seen that there are a number of levels of logging verbosity/detail that you can enable on each component.
These two lines of code enable debug logging at the INFO level for echo clients and servers. This will result in the application printing out messages as packets are sent and received during the simulation.

Now we will get directly to the business of creating a topology and running a simulation. We use the topology helper objects to make this job as easy as possible.

Topology Helpers

NodeContainer

The next two lines of code in our script will actually create the |ns3| Node objects that will represent the computers in the simulation.

```
NodeContainer nodes;
nodes.Create (2);
```

Let's find the documentation for the NodeContainer class before we continue. Another way to get into the documentation for a given class is via the Classes tab in the Doxygen pages.

If you still have the Doxygen handy, just scroll up to the top of the page and select the Classes tab. You should see a new set of tabs appear, one of which is Class List.

Under that tab you will see a list of all of the |ns3| classes. Scroll down, looking for ns3::NodeContainer.

When you find the class, go ahead and select it to go to the documentation for the class.

You may recall that one of our key abstractions is the Node.

This represents a computer to which we are going to add things like protocol stacks, applications and peripheral cards.

The NodeContainer topology helper provides a convenient way to create, manage and access any Node objects that we create in order to run a simulation.

The first line above just declares a **NodeContainer** which we call nodes. The second line calls the Create method on the nodes object and asks the container to create two nodes.

As described in the Doxygen, the container calls down into the |ns3| system proper to create two Node objects and stores pointers to those objects internally.

The nodes as they stand in the script do nothing. The next step in constructing a topology is to connect our nodes together into a network.

The simplest form of network we support is a single point-to-point link between two nodes. We'll construct one of those links here.

PointToPointHelper

We are constructing a point to point link, and, in a pattern which will become quite familiar to you, we use a topology helper object to do the low-level work required to put the link together.

Recall that two of our key abstractions are the NetDevice and the Channel.

In the real world, these terms correspond roughly to peripheral cards and network cables.

Typically these two things are intimately tied together and one cannot expect to interchange, for example, Ethernet devices and wireless channels.

Our Topology Helpers follow this intimate coupling and therefore you will use a single PointToPointHelper to configure and connect |ns3| PointToPointNetDevice and PointToPointChannel objects in this script.
The next three lines in the script are,

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue("2ms"));

The first line, PointToPointHelper pointToPoint;

instantiates a PointToPointHelper object on the stack. From a high-level perspective the next line,

pointToPoint.SetDeviceAttribute ("DataRate", StringValue("5Mbps"));

tells the PointToPointHelper object to use the value "5Mbps" (five megabits per second) as the "DataRate" when it creates a PointToPointNetDevice object.

From a more detailed perspective, the string "DataRate"
corresponds to what we call an Attribute of the PointToPointNetDevice.

If you look at the Doxygen for class ns3::PointToPointNetDevice and find the documentation for the GetTypeId method, you will find a list of Attributes defined for the device. Among these is the "DataRate" Attribute.

Most user-visible |ns3| objects have similar lists of Attributes. We use this mechanism to easily configure simulations without recompiling as you will see in a following section.
Similar to the "DataRate" on the PointToPointNetDevice you will find a "Delay" Attribute associated with the PointToPointChannel.

The final line,
pointToPoint.SetChannelAttribute ("Delay", StringValue("2ms"));
tells the PointToPointHelper to use the value "2ms" (two milliseconds) as the value of the propagation delay of every point to point channel it subsequently creates.

NetDeviceContainer

At this point in the script, we have a NodeContainer that contains two nodes.

We have a PointToPointHelper that is primed and ready to make PointToPointNetDevices and wire PointToPointChannel objects between them.