Practical - 01

Title: - Installation of NS-3 and NetAnim in Linux

Aim: - To install NS3 and NetAnim in Linux – Ubuntu 20

Description: -

ns-3 has been developed to provide an open, extensible network simulation platform, for networking research and education. In brief, ns-3 provides models of how packet data networks work and perform and provides a simulation engine for users to conduct simulation experiments. Some of the reasons to use ns-3 include to perform studies that are more difficult or not possible to perform with real systems, to study system behavior in a highly controlled, reproducible environment, and to learn about how networks work. Users will note that the available model set in ns-3 focuses on modeling how Internet protocols and networks work, but ns-3 is not limited to Internet systems; several users are using ns-3 to model non-Internet-based systems.

ns-3 is built as a system of software libraries that work together. User programs can be written that links with (or imports from) these libraries. User programs are written in either the C++ or Python programming languages.

ns-3 is distributed as source code, meaning that the target system needs to have a software development environment to build the libraries first, then build the user program

Operating system and compiler support

ns-3 is supported and currently tested on the following primary platforms:

1. Linux (x86 and x86_64): gcc/g++ versions 7 and above
    a. Note: If you are using RHEL or Centos, you will likely need to install a more up-to-date compiler than the default; search for how to enable 'software collections' or 'devtoolset' on these distributions. Other Linux distributions typically have a suitable default compiler (at least version 4.9).
2. MacOS Apple LLVM: version 11.0.0 and above
3. FreeBSD and Linux (x86_64): clang/LLVM version 8 and later

The minimum Python version supported is currently version 3.6 or greater (major version 3).

The following list of packages should be accurate through the Ubuntu 20.04 release; other releases or other Debian-based systems may slightly vary. Ubuntu 16.04 LTS release is probably the oldest release that is known to work with the most recent ns-3 releases.

Note: As of ns-3.30 release (August 2019), ns-3 uses Python 3 by default, but earlier releases depend on Python 2 packages, and at least a Python 2 interpreter is recommended. If working with an earlier release, one may in general substitute 'python' for 'python3' in the below (e.g. install 'python-dev' instead of 'python3-dev').

1) Download NS3 from the following website. It will be downloaded in downloads directory.

https://www.nsnam.org/releases/ns-3-32/

2) Perquisite for installing NS3.32

3) **sudo apt upgrade**

4) **sudo apt update**

5) minimal recommended requirements for release 3.30-3.35:

sudo apt install g++ python3

6) minimal requirements for Python API users (release 3.30 to ns-3.36): This is the minimal set of packages needed to work with Python bindings from a released tarball.

sudo apt install g++ python3 python3-dev pkg-config sqlite3 cmake

7) Netanim animator: qt5 development tools are needed for Netanim animator; qt4 will also work but we have migrated to qt5. qt6 compatibility is not tested.

sudo apt install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools

8) Support for ns-3-pyviz visualizer (release 3.36 and earlier)

sudo apt install gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3

9) Support for bake build tool:

sudo apt install mercurial unzip

10) Debugging

sudo apt install gdb valgrind

11) Doxygen and related inline documentation:

sudo apt install doxygen graphviz imagemagick

sudo apt install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils dvipng latexmk

12) 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):

sudo apt install python3-sphinx dia

13) To read pcap packet traces

sudo apt install tcpdump

14) Database support for statistics framework

sudo apt install sqlite sqlite3 libsqlite3-dev

15) Support for generating modified python bindings

sudo apt install cmake libc6-dev libc6-dev-i386 libclang-dev llvm-dev automake python3-pip

python3 -m pip install --user cxxfilt

16) After installing the required packages

Create a folder named workspace in the home directory and then put the NS3 tar downloaded from this official site (https://www.nsnam.org/releases/ns-3-32/) into the workspace folder.
Go to Home ⇒ right click ⇒ New Folder ⇒ named as workspace

17) Put your downloaded tar file in workspace folder:

18) Go to terminal and input these commands consecutively after each command finishes executing.

cd

cd workspace

tar -xvjf ns-allinone-3.32.tar.bz2

cd ns-allinone-3.32

19) Type next command to build:

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

Be patient and give time

20) After successfully build:

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

cd ns-3.32

./test.py

Numbers of passes can be different for different users

21) To check any application is running. do the following steps

cd ns-3.32/

./waf --run hello-simulator

NS-3 Simulator – Introduction

Many simulation tools exist for network simulation studies. Below are a few distinguishing features of ns-3 in contrast to other tools.

ns-3 is designed as a set of libraries that can be combined together and also with other external software libraries. While some simulation platforms provide users with a single, integrated graphical user interface environment in which all tasks are carried out, ns-3 is more modular in this regard. Several external animators and data analysis and visualization tools can be used with ns-3. However, users should expect to work at the command line and with C++ and/or Python software development tools.

ns-3 is primarily used on Linux or macOS systems, although support exists for BSD systems and also for Windows frameworks that can build Linux code, such as Windows Subsystem for Linux, or Cygwin. Native Windows Visual Studio is not presently supported although a developer is working on future support. Windows users may also use a Linux virtual machine.

ns-3 is not an officially supported software product of any company.

ns-3 is built as a system of software libraries that work together.

User programs can be written that links with (or imports from) these libraries.

User programs are written in either the C++ or Python programming languages.

ns-3 is distributed as source code, meaning that the target system needs to have a software development environment to build the libraries first, then build the user program.

Key Abstractions in NS3

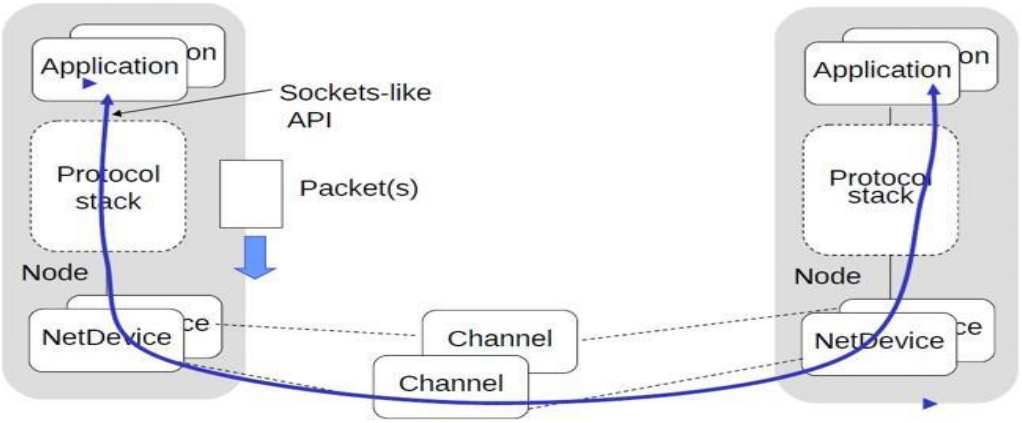
- **Node**
    - In Internet jargon, a computing device that connects to a network is called a host or sometimes an end system.
    - Because ns-3 is a network simulator, not specifically an Internet simulator, we intentionally do not use the term host since it is closely associated with the Internet and its protocols. Instead, we use a more generic term also used by other simulators that originates in Graph Theory — the node.
    - In ns-3 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 ns-3.

- **Application**
  - Typically, computer software is divided into two broad classes – System software and Application software
  - In ns-3 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.
  - ns-3 applications run on ns-3 Nodes to drive simulations in the simulated world.
  - In ns-3 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 ns-3, 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.
- **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.

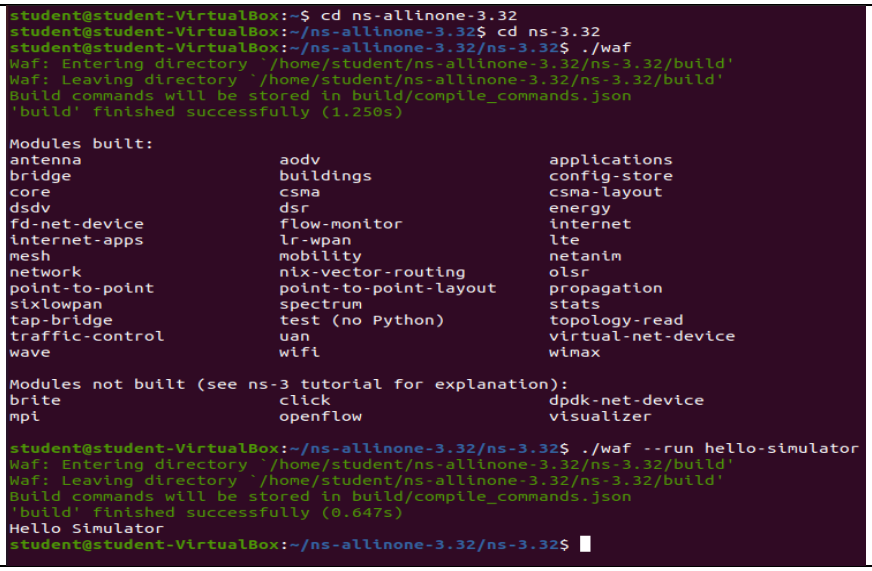
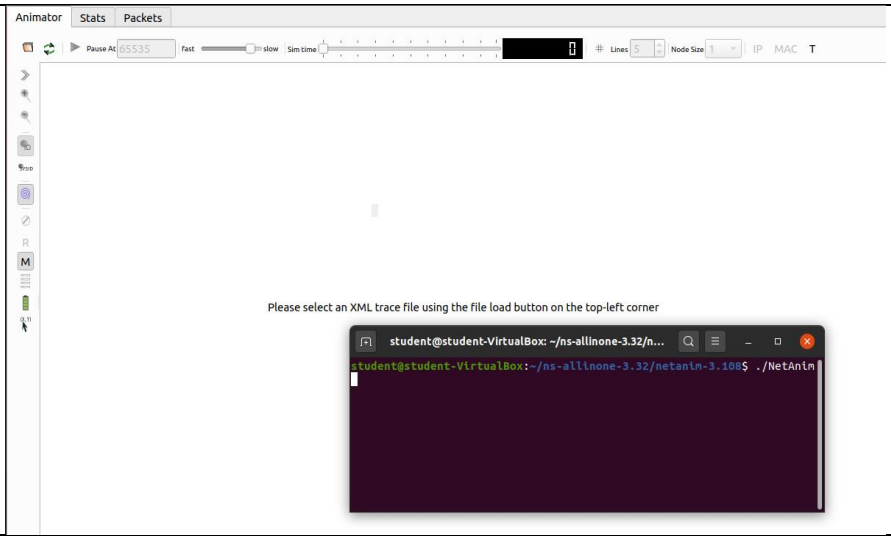
- Today most computers come with the network interface hardware built in and users don't see these building blocks.
- A NIC will not work without a software driver to control the hardware.
- In Unix (or Linux), a piece of peripheral hardware is classified as a device.
- Devices are controlled using device 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 ns-3 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.

◈ **Topology Helpers**

- In a real network, you will find host computers with added (or built-in) NICs.
- In ns-3 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 ns-3, we provide what we call topology helpers to make this as easy as possible.
- For example, it may take many distinct ns-3 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.



Exercises

1	Install NS3 in Ubuntu and execute hello-simulator script.
Output	
2	Install Netanim tool on Ubuntu and check if its properly installed.
Output	

```
student@student-VirtualBox:~$ cd ns-allinone-3.32
student@student-VirtualBox:~/ns-allinone-3.32$ cd ns-3.32
student@student-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf
Waf: Entering directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.250s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     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
wave                     wifi                     wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    dpdk-net-device
mpi                      openflow                 visualizer

student@student-VirtualBox:~/ns-allinone-3.32/ns-3.32$ ./waf --run hello-simulator
Waf: Entering directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/student/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.647s)
Hello Simulator
student@student-VirtualBox:~/ns-allinone-3.32/ns-3.32$
```


Conclusion: Installed NS3 and NetAnim tool on Ubuntu.