EXPERIMENT NO. 9

Aim: Study and Installation of Network Simulator (NS3).

Requirements:

Software	**Package/version**
Linux O.S	Any new variant distro (Arch, Debian, Fedro, etc.)
C++ compiler	clang++ or g++ (g++ version 4.9 or greater)
Python Git	python2 version >= 2.7.10, or python3 version >=3.4 any recent version
tar	any recent version
bunzip2	any recent version

Theory:

Introduction to NS3:

Network simulator is a tool used for simulating the real world network on one computer by writing scripts in C++ or Python. Normally if we want to perform experiments, to see how our network works using various parameters. We don't have required number of computers and routers for making different topologies. Even if we have these resources it is very expensive to build such a network for experiment purposes.

So to overcome these drawbacks we used NS3, which is a discrete event network simulator for Internet. NS3 helps to create various virtual nodes (i.e., computers in real life) and with the help of various Helper classes it allows us to install devices, internet stacks, application, etc. to our nodes.

Using NS3 we can create Point-to-point, Wireless, CSMA, etc. connections between nodes. Point-to-point connection is same as a LAN connected between two computers. Wireless connection is same as Wi-Fi connection between various computers and routers. CSMA connection is same as bus topology between computers. After building connections we try to install NIC to every node to enable network connectivity.

When network cards are enabled in the devices, we add different parameters in the channels (i.e., real world path used to send data) which are data-rate, packet size, etc. Now we use Application to generate traffic and send the packets using these applications.

NS3 gives us special features which can be used for real life integrations. Some of these features are:

Tracing of the nodes:

NS3 allows us to trace the routes of the nodes which helps us to know how much data is send or received. Trace files are generated to monitor these activities.

NetAnim:

It stands for Network Animator. It is an animated version of how network will look in real and how data will be transferred from one node to other.

Pcap file:

NS3 helps to generate pcap file which can be used to get all information of the packets (e.g., Sequence number, Source IP, destination IP, etc.). These pcaps can be seen using a software tool known as Wireshark.

Gnu Plot:

Gnu Plot is used to plot graphs from the data which we get from trace file of NS3. Gnu plot gives more accurate graph compare to other graph making tools and also it is less complex than other tools.

Advantages:

- The system has been modularized
- To allow for modular libraries
- Individual modules contains with directory structure
- To allow the node to use external routing

Disadvantages:

- Ns3 suffers from lack of credibility
- Modules, component based on ns2
- Ns3 needs lot of maintainers
- Active maintainers are required

Introduction to NS2:

NS-2 can be used to implement network protocols such as TCP and UPD, traffic source behaviour such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms and many more. In ns2, C++ is used for detailed protocol implementation and Otcl is used for the setup. The compiled C++ objects are made available to the Otcl interpreter and in this way, the ready-made C++ objects can be controlled from the Otcl level.

Downloading NS3:

This option is for the new user who wishes to download and experiment with the most recently released and packaged version of NS3. NS3 publishes its releases as compressed source archives, sometimes referred to as a tarball. A tarball is a particular format of software archive where multiple files are bundled together and the archive is usually compressed. The process for downloading NS3 via tarball is simple; you just have to pick a release, download it and uncompress it.

Let’s assume that you, as a user, wish to build NS3 in a local directory called workspace. If you adopt the workspace directory approach, you can get a copy of a release by typing the following into your Linux shell (substitute the appropriate version numbers, of course)

```
$ cd
$ mkdir workspace
$ cd workspace
$ wget https://www.nsnam.org/release/ns-allinone-3.29.tar.bz2
$ tar xjf ns-allinone-3.29.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-3.29, you should see a number of files and directories

```
$ cd ns-allinone-3.29
$ ls
bake      constants.py   NS3.29                          README
build.py  netanim-3.108  pybindgen-0.17.0.post58+ngcf00cc0  util.py
```

You are now ready to build the base NS3 distribution.

Building NS3:

When working from a released tarball, a convenience script available as part of NS3-allinone can orchestrate a simple build of components. This program is called build.py. This program will get the project configured for you in the most commonly useful way. However, please note that more advanced configuration and work with NS3 will typically involve using the native NS3 build system, Waf, to be introduced later in this tutorial.

If you downloaded using a tarball you should have a directory called something like ns-allinone-3.29 under your ~/workspace directory. Type the following:

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

Because we are working with examples and tests in this tutorial, and because they are not built by default in NS3, the arguments for build.py tells it to build them for us. The program also defaults to building all available modules. Later, you can build NS3 without examples and tests, or eliminate the modules that are not necessary for your work, if you wish.

You will see lots of compiler output messages displayed as the build script builds the various pieces you downloaded. First, the script will attempt to build the netanim animator, then the pybindgen bindings generator, and finally NS3. Eventually you should see the following:

```
Waf: Leaving directory '/path/to/workspace/ns-allinone-3.29/NS3.29/build'
'build' finished successfully (6m25.032s)


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               mpi
netanim (no Python)   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 NS3 tutorial for explanation):
brite                 click                  openflow

Leaving directory ./NS3.29
```

Conclusion: We have successfully learnt the concept of NS3 and installed it in our Linux O.S using source archive release of NS3.