

Technical Note

Deployment of NS-3 with Eclipse IDE

José Vega-Sánchez¹, Lesly Maygua-Marcillo¹, Luis Urquiza-Aguilar^{1,*} and Pablo Barbecho-Bautista²

¹ Departamento de Electrónica, Telecomunicaciones y Redes de Información, Facultad de Eléctrica y Electrónica, Escuela Politécnica Nacional (EPN), C. Ladrón de Guevara E11-253, Quito PO.Box 17-01-2759, Ecuador; jose.vega01@epn.edu.ec, lesly.maygua@epn.edu.ec

² Departamento de Ingeniería Telemática, Universidad Politécnica de Catalunya (UPC). España; pablo.barbecho@estudiant.upc.edu

* Correspondence: luis.urquiza@epn.edu.ec; Tel.: +593 2297-6300 ext:2311

Abstract: Network Simulators is typically used to study services and applications in complex scenarios due to the infeasibility of deploying real testbeds. Many problems can be solved by using network simulators such as NS-3. With this in mind, the aim of this article is to introduce new NS-3 users through detailed information. It is sometimes difficult to handle by new users the traditional manuals developed by NS-3 project official website. In this article, NS-3 for communication network and Eclipse Integrated Development Environment (IDE) for powerful programming language are integrated step-by-step, explaining the main features of these open source software packages and concluding with an example simulation. Our effort is to make it easy for a beginner to be part of the NS-3 research community and to maintain an open environment of knowledge.

Keywords: Network simulators, NS-3, simulation, network

1. Introduction

Nowadays, communications are present in everything. The different applications allow to connect organizations, universities, colleges, hospitals, vehicles and even countries. Metrics such as, throughput, delay, number of retransmissions, jitter or packet error rate are essential for assessing and designing communication networks. For this purpose, simulation is the most utilized method to study networks, it is the imitation of the real situations through tools called simulators, such as: NS-2, NS-3, OPNET [1], NetSim, OMNeT++, among others [2]. Network simulators predicts the behavior of the different networks e.g., allow the study of wired networks, wireless networks [3] [4], mobile telephony [5], QoS analysis [6], protocols analysis [7] [8], etc.

In this context, the new generation of network simulator NS-3 includes a more accurate physical (PHY) layer metrics [9] [10]. The NS-3 started in 2006, is an open source software which is written from C++ and is not an evolution of NS-2, it is a new simulator. On the other hand, several external animators and visualization tools can be used with NS-3 such as C++ and/or Python packages [11]. This article makes use of Eclipse IDE C++ as integration tool on NS-3. Eclipse is widely used in computer programming. Although, the Eclipse is written primarily in Java, it supports plug-ins that allow developers to develop applications in different programming languages (e.g. PHP, Perl, Cobol and Python) [12]. Integrated graphical user interface on NS-3 is a good idea for improving performance of new generation of simulators. The article is organized as follows: Section 2 is focused in the Installation of NS-3 and Eclipse CDT. The Installation of Java Development Kit (JDK) is derived in Section 3. Finally, Section 4 deals with the Installation and Configure Eclipse for C++ Developers for running the first script in NS-3 on Eclipse IDE.

2. Installation of NS-3 and Eclipse CDT

2.1. Prerequisite Package for Ubuntu

First of all, it is advisable to update the Ubuntu 16.04 with its latest components through the following instructions in a terminal. Remember you need "sudo" operation to run the commands as **root** user, you will be prompted for your password only once.

1	sudo apt-get update
2	sudo apt-get upgrade
3	sudo apt-get clean

Figure 1. Prerequisite Package for Ubuntu.

Now, the core of NS-3 requires a list of packages that are needed to support different NS-3 options, as detailed below [13]:

Minimal Requirements for Python

- gcc, g++, python, python-dev, mercurial, bzip2.

Debugging and GNU Scientific Library (GLS) support

- gdb, valgrind, gsl-bin, libgsl0-dev, libgsl2, libgsl2:i386.

Requirements for Network Simulation Cradle

- flex, bison.

Reading pcap packet traces

- tcpdump.

Database support for statistics framework

- sqlite, sqlite3, libsqlite3-dev.

Xml-based version of the config store

- libxml2, libxml2-dev.

A GTK-based configuration system

- libgtk2.0-0, libgtk2.0-dev.

Support for utils/check-style.py code style check program

- uncrustify.

Doxygen and related inline documentation

- doxygen, graphviz, imagemagick, texlive, texlive-latex-extra, texlive-generic-extra, texlive-generic-recommended, texinfo, dia, texlive-extra-utils, texi2html.

Support for Gustavo Carneiro's NS-3-pyviz visualizer

- python-pygraphviz, python-kiwi, python-pygoocanvas, libgoocanvas-dev, python-pygccxml, python-gnome2, python-gnomedesktop, python-rsvg, ipython.

In short, provide the following commands in a terminal for the installation of these prerequisites which were described above:

```
sudo apt-get install gcc g++ python python-dev mercurial bzip2 gdb valgrind gsl-bin libgsl0-dev libgsl2 libgsl2:i386 flex bison tcpdump sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick texlive texlive-latex-extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive texlive-latex-extra texlive-extra-utils texlive-generic-recommended texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev python-pygccxml
```

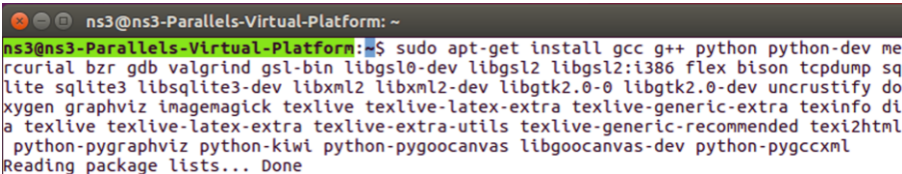


Figure 2. Prerequisite Package for NS-3.

After pressing the ENTER button, you should see something like the following:

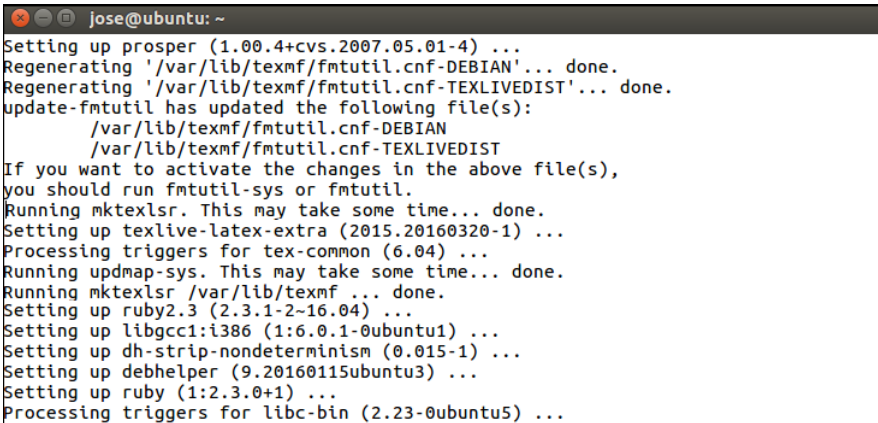


Figure 3. Successful installation of Prerequisite Package for NS-3.

2.2. Download and Install NS-3

With the previous step completed, create a directory of your preference in the home folder (e.g. in my case ns3). After that, extract the downloaded NS-3 package in the new location using the following commands from a terminal:

1	cd /home/user_name
2	mkdir ns3
3	cd ns3
4	wget https://www.nsnam.org/release/ns-allinone-3.27.tar.bz2
5	tar xjf ns-allinone-3.27.tar.bz2
6	cd ns-allinone-3.27/

Figure 4. Commands to Download the NS-3 package.

Once download and extracted the NS-3 package, is time to compile and build the default examples. You should be patient; the compilation takes several minutes. Then, run the command below:

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

Figure 5. Command to build and compile the examples of NS-3.

If the build is successful, the message “build finished successfully ” will appear

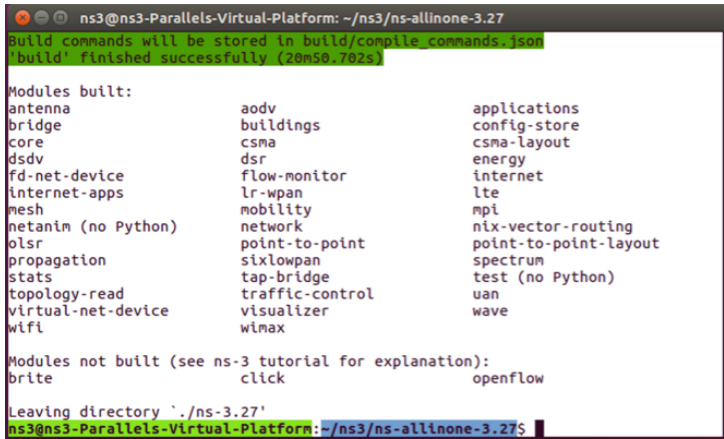


Figure 6. Confirmation message of successful compilation.

At this point, it is necessary to build using **waf** builder by running the commands below:

```
1 cd ns-3.27
2 ./waf -d debug --enable-examples --enable-tests configure
```

Figure 7. Command to build the examples of NS-3 with waf tool.

If all is ok, you will get this message "configure finished successfully"

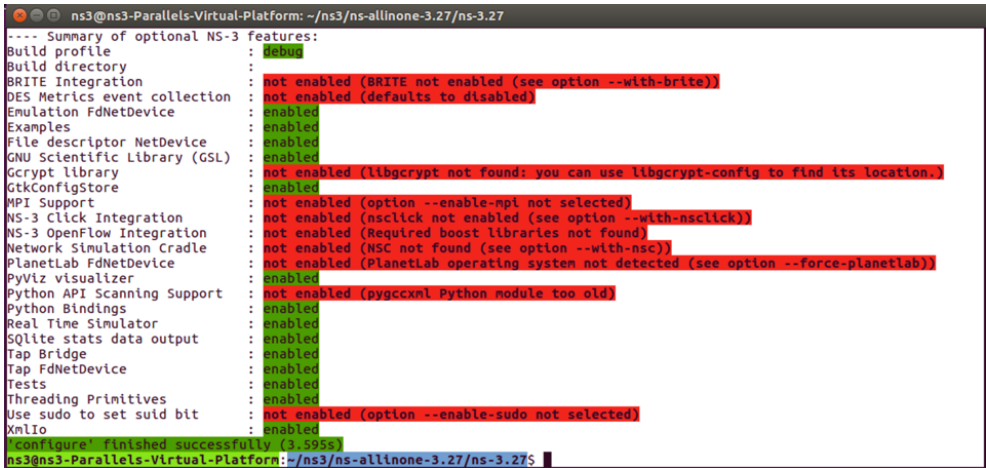


Figure 8. Confirmation message of successful compilation.

It is time to check if the installation is successfully installed by running python test file. Type the following command:

1	./test.py
---	-----------

Figure 9. Command to test if everything is alright.

If all modules passed, you will get the following output. Congratulations, NS-3 is successfully installed.

```
ns3@ns3-Parallels-Virtual-Platform: ~/ns3/ns-allinone-3.27/ns-3.27
owerB=15 --delay=10 --txModeA=OfdmRate6Mbps --txModeB=OfdmRate6Mbps --checkResults=1 --expect
RxASuccessfull=0 --expectRxBSuccessfull=1
PASS: Example src/wimax/examples/wimax-simple
PASS: Example src/wimax/examples/wimax-ipv4
PASS: Example src/wimax/examples/wimax-multicast
PASS: Example examples/tutorial/first.py
PASS: Example examples/wireless/wifi-ap.py
PASS: Example src/wifi/examples/wifi-manager-example --wifiManager=Ideal --standard=802.11ax-
2.4GHz --serverChannelWidth=20 --clientChannelWidth=20 --serverShortGuardInterval=800 --clien
tShortGuardInterval=800 --serverNss=2 --clientNss=2 --stepTime=0.1
PASS: Example examples/routing/simple-routing-ping6.py
PASS: Example src/bridge/examples/csma-bridge.py
PASS: Example src/core/examples/sample-simulator.py
PASS: Example examples/wireless/mixed-wired-wireless.py
PASS: Example src/flow-monitor/examples/wifi-olsr-flowmon.py
554 of 557 tests passed (554 passed, 3 skipped, 0 failed, 0 crashed, 0 valgrind errors)
List of SKIPped tests:
  ns3-tcp-cwnd
  ns3-tcp-interoperability
  nsc-tcp-loss
ns3@ns3-Parallels-Virtual-Platform:~/ns3/ns-allinone-3.27/ns-3.27$
```

Figure 10. Successful installation of NS-3.

Finally, you only need to run a “hello word” program by typing the following:

1	cd ns3/ns-allinone-3.27/ns-3.27/
2	./waf --run examples/tutorial/hello-simulator

Figure 11. Running the first script.

The above commands produce the following output:

```
ns3@ns3-Parallels-Virtual-Platform: ~/ns3/ns-allinone-3.27/ns-3.27
ns3@ns3-Parallels-Virtual-Platform:~/ns3/ns-allinone-3.27/ns-3.27$ sudo ./waf --run
examples/tutorial/hello-simulator
[sudo] password for ns3:
Waf: Entering directory '/home/ns3/ns3/ns-allinone-3.27/ns-3.27/build'
Waf: Leaving directory '/home/ns3/ns3/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.326s)
Hello Simulator
ns3@ns3-Parallels-Virtual-Platform:~/ns3/ns-allinone-3.27/ns-3.27$
```

Figure 12. Output message of the first script.

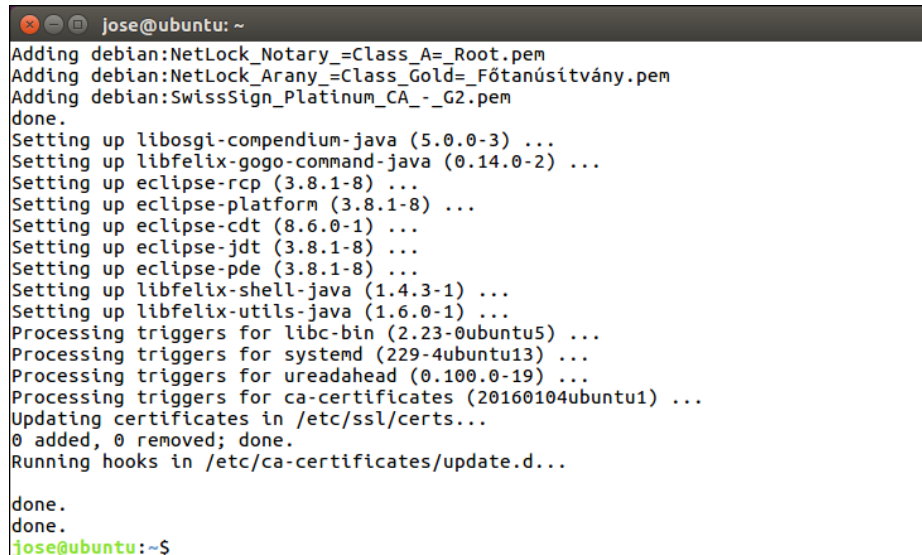
2.3. Install Eclipse CDT

It is worth mentioning that there are two ways to install Eclipse CDT: either running the commands of installation from a terminal, or using the Synaptic package manager. In this article, we choose the first option, because in the second alternative, the synaptic package needs to be installed and in some cases you may experience some errors in this installation process. Having said that, open a terminal and execute the following command:

```
1 sudo apt-get install eclipse-cdt
```

Figure 13. Command to install the Eclipse CDT

The image below shows how it looks upon successful execution

A terminal window titled 'jose@ubuntu: ~' showing the output of the command 'sudo apt-get install eclipse-cdt'. The output lists various packages being installed, including libosgi-compndium-java, libfelix-gogo-command-java, eclipse-rcp, eclipse-platform, eclipse-cdt, eclipse-jdt, eclipse-pde, libfelix-shell-java, and libfelix-utils-java. It also shows the processing of triggers for libc-bin, systemd, ureadahead, and ca-certificates, and the updating of certificates in /etc/ssl/certs. The installation completes successfully with 'done.' and 'jose@ubuntu:~\$' at the prompt.

```
jose@ubuntu: ~
Adding debian:NetLock_Notary_=Class_A=_Root.pem
Adding debian:NetLock_Arany_=Class_Gold=_Főtanúsítvány.pem
Adding debian:SwissSign_Platinum_CA_-_G2.pem
done.
Setting up libosgi-compndium-java (5.0.0-3) ...
Setting up libfelix-gogo-command-java (0.14.0-2) ...
Setting up eclipse-rcp (3.8.1-8) ...
Setting up eclipse-platform (3.8.1-8) ...
Setting up eclipse-cdt (8.6.0-1) ...
Setting up eclipse-jdt (3.8.1-8) ...
Setting up eclipse-pde (3.8.1-8) ...
Setting up libfelix-shell-java (1.4.3-1) ...
Setting up libfelix-utils-java (1.6.0-1) ...
Processing triggers for libc-bin (2.23-0ubuntu5) ...
Processing triggers for systemd (229-4ubuntu13) ...
Processing triggers for ureadahead (0.100.0-19) ...
Processing triggers for ca-certificates (20160104ubuntu1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
jose@ubuntu:~$
```

Figure 14. Successful installation of Eclipse CDT

3. Install Java Development Kit (JDK)

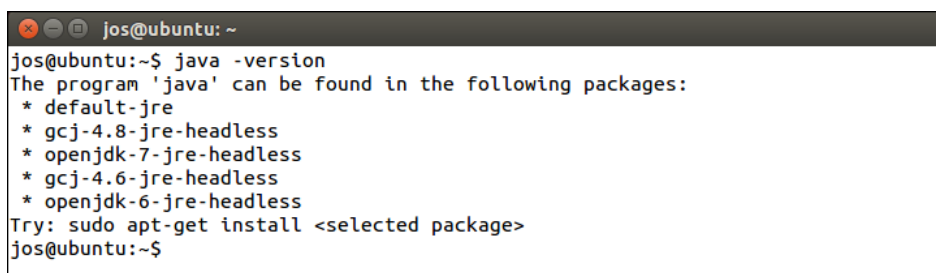
3.1. Prerequisite Package for Java

As a first step, it is necessary to check the java version in Ubuntu. For this, in a terminal use the following command:

```
1 sudo java -version
```

Figure 15. Command to check the Java version in Ubuntu.

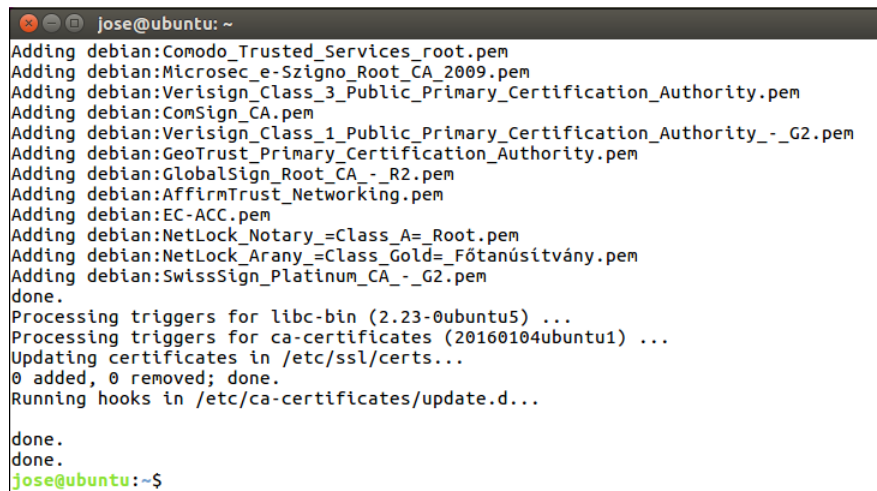
If you do not have any java version installed, you should see output that looks similar to the following:

A terminal window titled 'jos@ubuntu: ~' showing the output of the command 'java -version'. The output states that the program 'java' can be found in the following packages: default-jre, gcj-4.8-jre-headless, openjdk-7-jre-headless, gcj-4.6-jre-headless, and openjdk-6-jre-headless. It suggests using 'sudo apt-get install <selected package>' to install one of them. The prompt returns to 'jos@ubuntu:~\$'.

```
jos@ubuntu: ~
jos@ubuntu:~$ java -version
The program 'java' can be found in the following packages:
* default-jre
* gcj-4.8-jre-headless
* openjdk-7-jre-headless
* gcj-4.6-jre-headless
* openjdk-6-jre-headless
Try: sudo apt-get install <selected package>
jos@ubuntu:~$
```

Figure 16. Prerequisite Package for Java JDK.

Now, install all packages required by java (e.g. `sudo apt-get install default-jre`). Afterwards, you should see something like the following:



```
jose@ubuntu: ~
Adding debian:Comodo_Trusted_Services_root.pem
Adding debian:Microsec_e-Szigno_Root_CA_2009.pem
Adding debian:Verisign_Class_3_Public_Primary_Certification_Authority.pem
Adding debian:ComSign_CA.pem
Adding debian:Verisign_Class_1_Public_Primary_Certification_Authority_-_G2.pem
Adding debian:GeoTrust_Primary_Certification_Authority.pem
Adding debian:GlobalSign_Root_CA_-_R2.pem
Adding debian:AffirmTrust_Networking.pem
Adding debian:EC-ACC.pem
Adding debian:NetLock_Notary_Class_A_Root.pem
Adding debian:NetLock_Arany_Class_Gold_Főtanúsítvány.pem
Adding debian:SwissSign_Platinum_CA_-_G2.pem
done.
Processing triggers for libc-bin (2.23-0ubuntu5) ...
Processing triggers for ca-certificates (20160104ubuntu1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
done.
jose@ubuntu:~$
```

Figure 17. Successful installation of the prerequisite Package for Java.

3.2. Install Oracle Java JDK 8

For this tutorial, the JDK 8 package is the last available version in the official Java website <http://tipsonubuntu.com/2016/07/31/install-oracle-java-8-9-ubuntu-16-04-linux-mint-18/>. Once the Java version is chosen, download Oracle through the following line in a terminal:

```
1 sudo add-apt-repository ppa:webupd8team/java
```

Figure 18. Command to download the Oracle Java.

If the download is successful, you should see the following output. Also, you will be asked if you want to continue, press [ENTER].



```
ns3@ns3-Parallels-Virtual-Platform: ~/ns3/ns-allinone-3.27/ns-3.27
Oracle Java 9 (For both Ubuntu and Debian): http://www.webupd8.org/2015/02/install-oracle-java-9-in-ubuntu-linux.html

Oracle JDK 9 is now considered stable. There are currently only 64bit builds (no other builds are available for download: http://www.oracle.com/technetwork/java/javase/download/s/index.html )
More info: https://launchpad.net/~webupd8team/+archive/ubuntu/java
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmp1lvhj9p/secring.gpg' created
gpg: keyring `/tmp/tmp1lvhj9p/pubring.gpg' created
gpg: requesting key EEA14886 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmp1lvhj9p/trustdb.gpg: trustdb created
gpg: key EEA14886: public key "Launchpad VLC" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg: imported: 1 (RSA: 1)
OK
ns3@ns3-Parallels-Virtual-Platform: ~/ns3/ns-allinone-3.27/ns-3.27$
```

Figure 19. Message for downloading Java.

After that, it's necessary check updates in the repositories using:`sudo apt-get update`.

Although Oracle JDK 9 is now considered stable, it is not compatible with Eclipse. We have to type the following command to install Oracle Java 8:

```
1 sudo apt-get install oracle-java8-installer
```

Figure 20. Command to install Oracle Java.

Next step, you must accept the license terms by pressing [Ok] and [Yes]

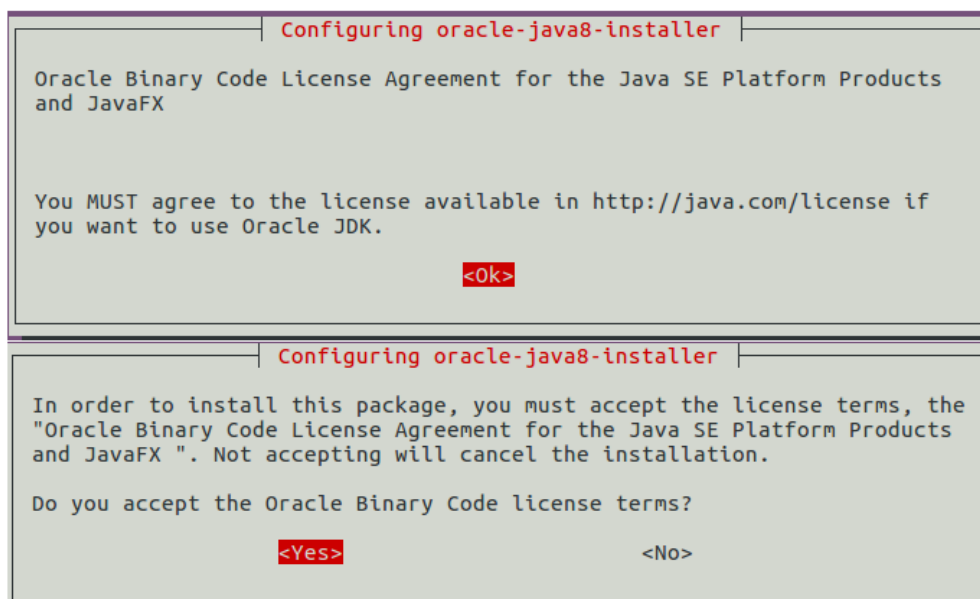


Figure 21. Acceptance of the License Agreement.

If you find below output it means your installation is successful.

```
jose@ubuntu: ~
usr/bin/serialver (serialver) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-oracle/bin/wsgen to provide /usr/
bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-oracle/bin/wsimport to provide /u
sr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-oracle/bin/xjc to provide /usr/bi
n/xjc (xjc) in auto mode
update-binfmts: warning: current package is openjdk-9, but binary format already
installed by openjdk-8
update-alternatives: using /usr/lib/jvm/java-8-oracle/jre/lib/amd64/libnpjp2.so
to provide /usr/lib/mozilla/plugins/libjavaplugin.so (mozilla-javaplugin.so) in
auto mode
Oracle JRE 8 browser plugin installed
Oracle JDK 8 installed

#####Important#####
To set Oracle JDK8 as default, install the "oracle-java8-set-default" package.
E.g.: sudo apt install oracle-java8-set-default
On Ubuntu systems, oracle-java8-set-default is most probably installed
automatically with this package.
#####

Setting up oracle-java8-set-default (8u121-1-webupd8~0) ...
jose@ubuntu:~$
```

Figure 22. Successful installation of Oracle Java.

Once the installation has been completed, the command of Fig. 15 produces the following output:

```
jose@ubuntu: ~
jose@ubuntu:~$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
jose@ubuntu:~$
```

Figure 23. Installed version of Java in Ubuntu.

4. Install and Configure Eclipse IDE for C/C++ Developers

4.1. Download and Install Eclipse IDE

Before installing Eclipse IDE, you should check if your operative system (OS) is 32bit or 64bit version by using:**uname -m**. With this information, download the suitable package for your Ubuntu version from website <https://eclipse.org/downloads/>. Later, unpack the Eclipse package and install it by using the commands from a terminal:

1	cd Downloads/
2	tar -xvzf eclipse-inst-linux64.tar.gz
3	cd eclipse-installer/
4	./eclipse-inst

Figure 24. Commands to download and install Eclipse IDE.

A window will open to you. Choose "**Eclipse IDE for C/C++ Developers**"

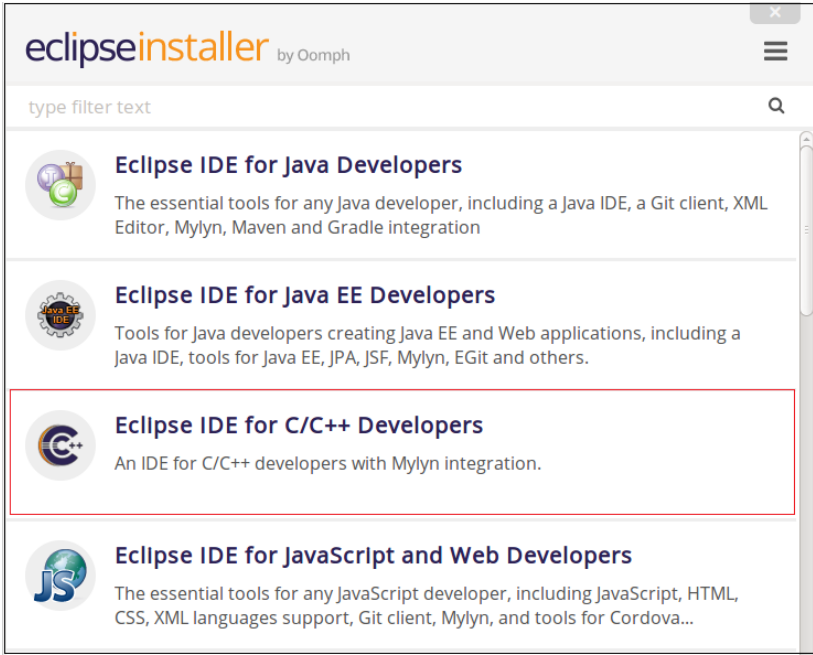


Figure 25. Available versions of Eclipse IDE.

next step, click on **"Install"**. It will take a few minutes.

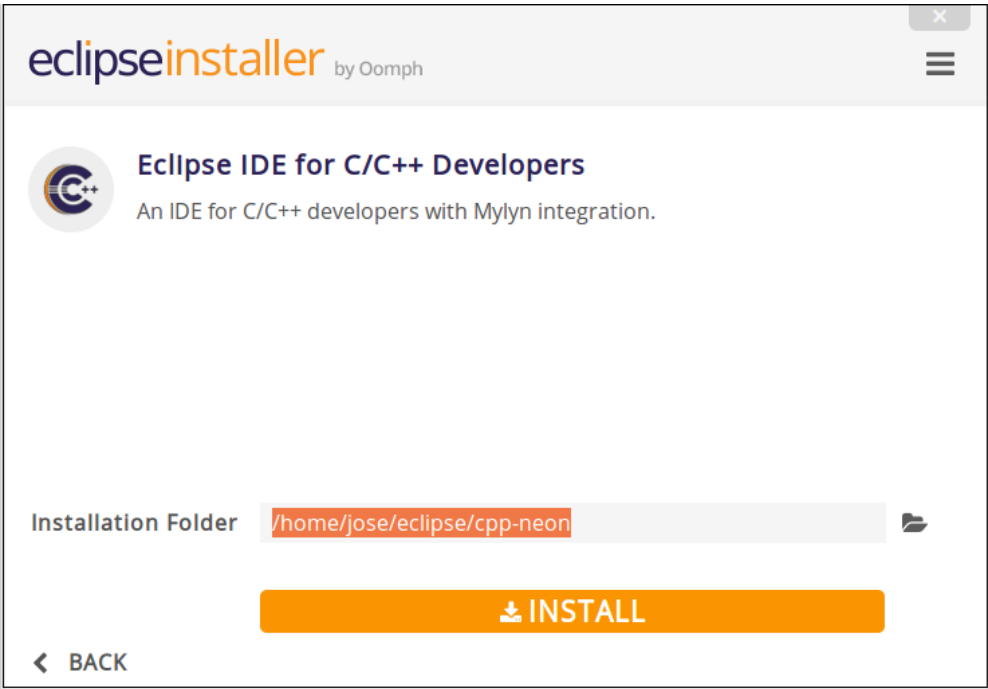


Figure 26. Eclipse IDE installer.

Then, accept the license terms, and then wait to see if package is installed successfully. Afterwards, click on **"Launch"** and Eclipse IDE will open to you.

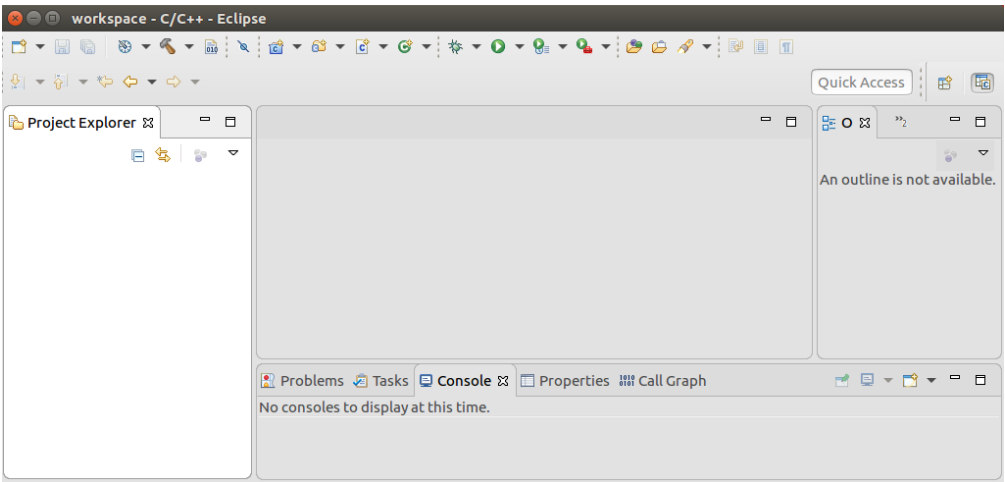


Figure 27. Eclipse IDE interface.

4.2. *Configure Eclipse Setting for NS-3 Environment*

4.2.1. Install MercurialEclipse

To achieve a friendly environment between Eclipse and NS-3, MercurialEclipse plugin needs to be installed for efficient project management. There are two ways to install MercurialEclipse, in this

tutorial both techniques are explained. In addition, it is shown the pros and cons of each of these options.

Option 1

This alternative has a major drawback, the MercurialEclipse installation package is unavailable in the official Eclipse Studio website, this issue is shown below. On the other hand, for this option, use the following instructions:

- * In the main menu go to **Help** → **Install New Software**
- * In "Work with" textbox enter the following website <http://cbes.javaforge.com/update> and clicking on "Add".
- * A new window will open to you. In "Name" textbox enter again the website <http://cbes.javaforge.com/update> and pressing the "OK" button.

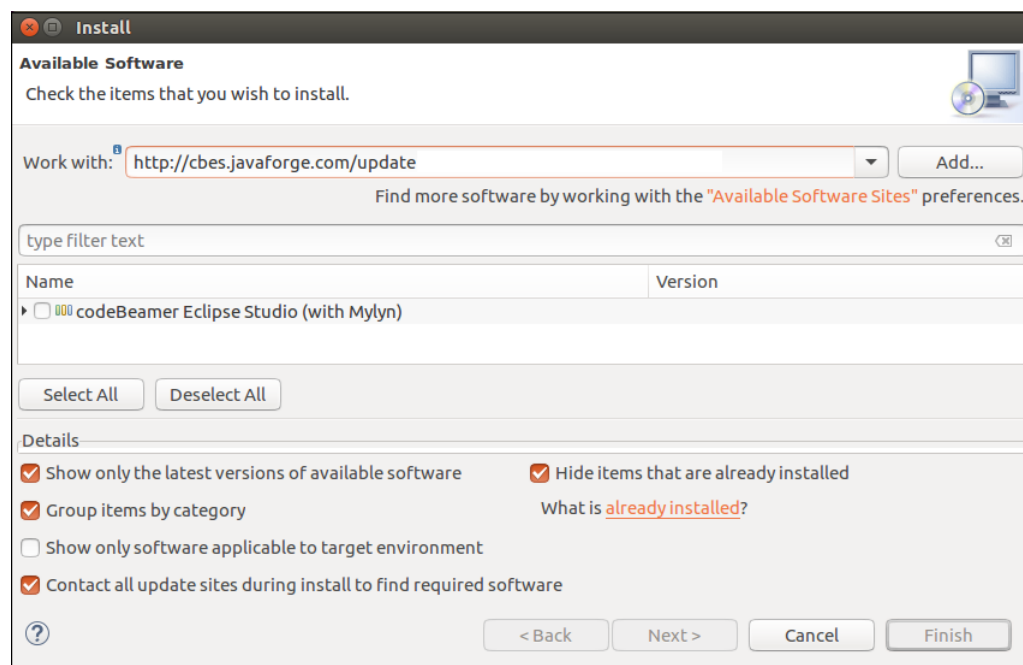


Figure 28. MercurialEclipse installer-Option 1.

As previously mentioned, the only available repository in the Fig. 28 is the **codeBeamer Eclipse Studio** package. Therefore, we recommend using option 2.

Option 2

This alternative is an user-friendly application, but the **Eclipse Marketplace** tool needed for this option is only available in the latest versions of Eclipse IDE. However, this article provides the installation instructions for latest portable eclipse, so you will not have any problem. Then, follow the instructions for this choice:

- * In the main menu go to **Help** → **Eclipse Marketplace**
- * Click on **Search** tab and write "MercurialEclipse"
- * Click on **"Install"**.

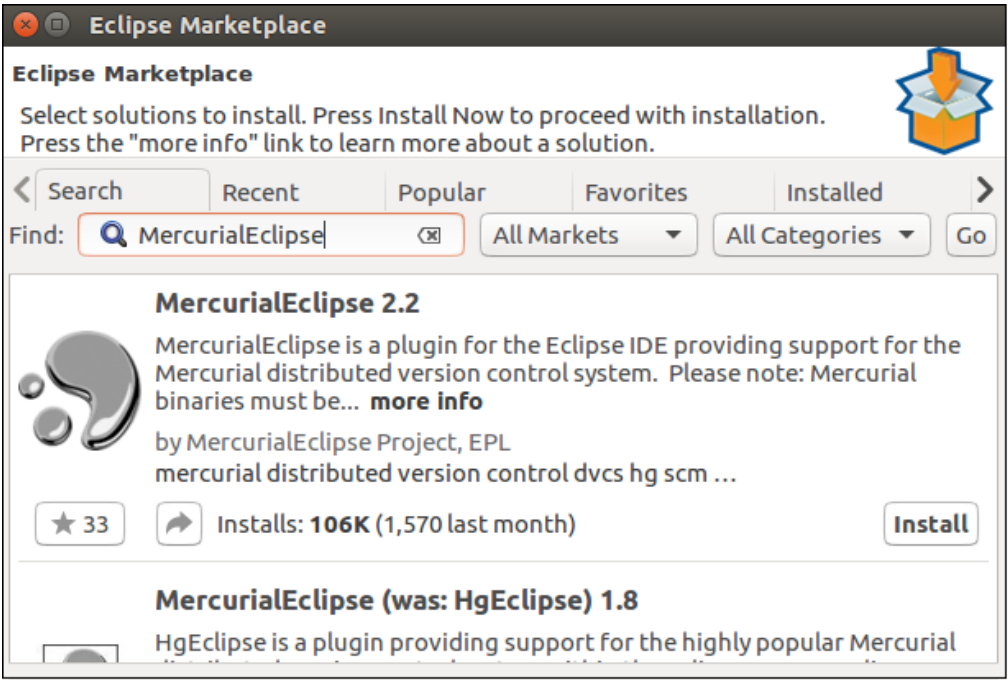


Figure 29. MercurialEclipse installer-Option 2.

then, accept the license terms and press the Finish button

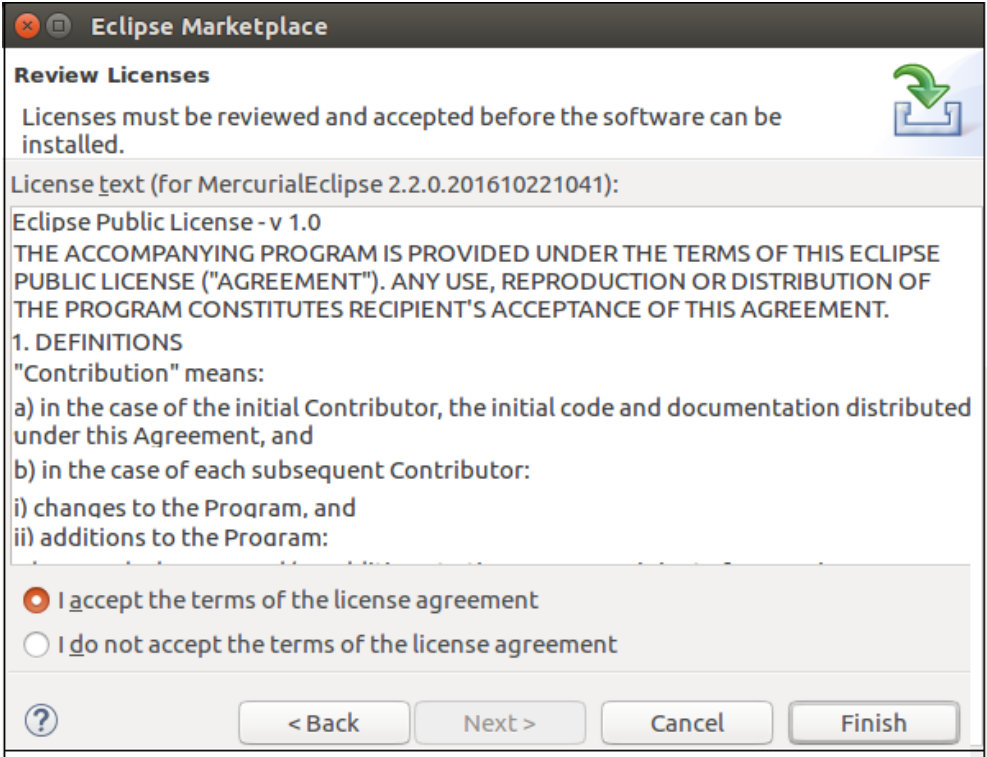


Figure 30. Acceptance of License Agreement for MercurialEclipse.

4.2.2. Create a New C++ Project

Now, it is time to create the first C++ project with the necessary configurations to link NS-3 with Eclipse IDE. To this end, follow the instructions:

- * Go to **File** → **New** → **C++ Project**

Fill the settings with the information below:

- * Choose a project name
- * Select Cross GCC compiler
- * Select NS-3's folder as location
- * Click on **"Next"**

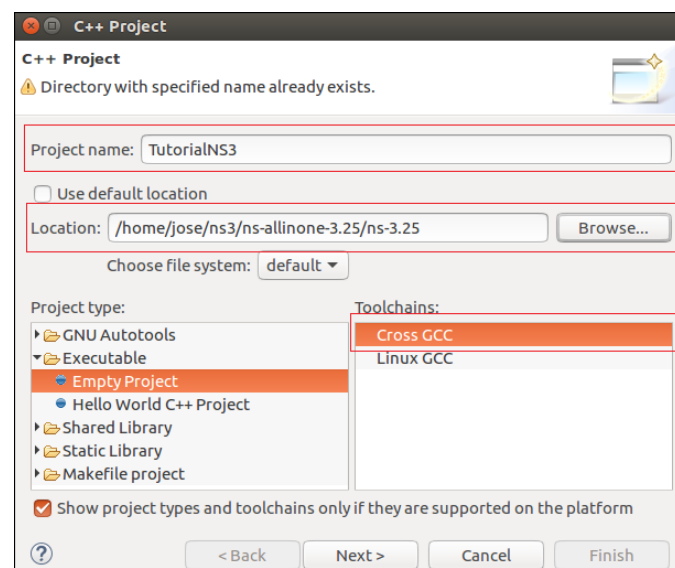


Figure 31. Configurations for the new C++ Project.

Next step, click on **"Next"** and **"Finish"**. The new C++ project was created:

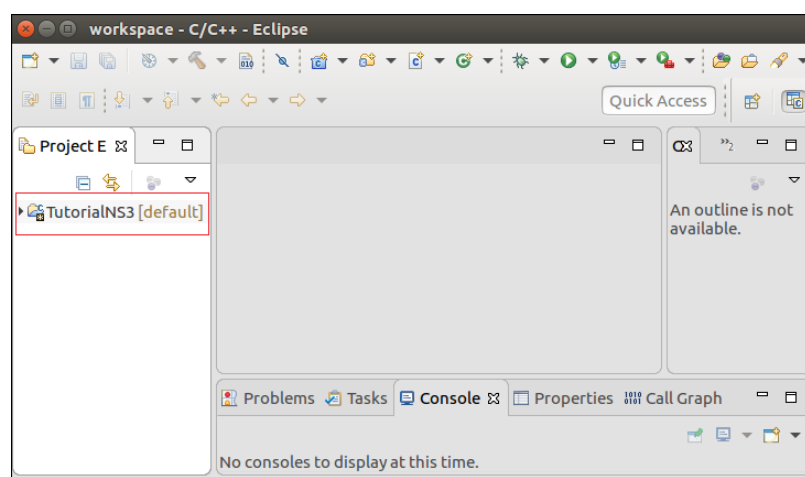


Figure 32. Eclipse IDE interface.

Share the project, To do that: Right click on the C++ project → **Team** → **share project**. In the new window choose the "Mercurial" option and click on **"Next"**

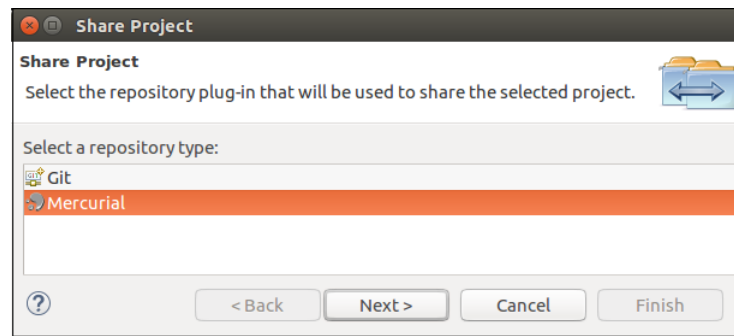


Figure 33. Mercurial plugin selection.

Confirm the Mercurial Repository Location and press the “**Finish**” button

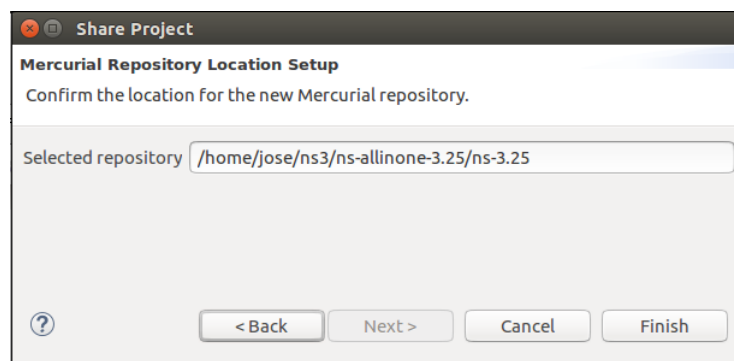


Figure 34. Mercurial plugin location.

4.2.3. Configure WAF Builder

Waf is a piece of software written in Python and used to help building software projects [14]. As shown in Fig. 11, the setting of this tool is essential for running the NS-3 scripts. Knowing this, follow the instructions:

- * Right click on the C++ project → **Properties** → **C/C++ Build**.
- * Uncheck the box Use default build command
- * Uncheck the box Generate makefile automatically
- * In “Build command” write the location to waf, in my case
“/home/jose/ns3/ns-allinone-3.25/ns-3.25/waf”
- * In “Build directory” write the location to build, in my case
“/home/jose/ns3/ns-allinone-3.25/ns-3.25/build”

Now, switch to Behavior tab and change the word "all" to "build" in the box incremental build

4.2.4. Configure the debugger

This setting aims to debug a test application under the scratch folder. Here are the steps:

- * In the main menu go to Run → **Debug Configurations**
- * Click on C/C++ Application and choose the C++ project created, in my case “TutorialNS3Debug”
- * Click on the “Browse” button under the “Project” section and select the current project

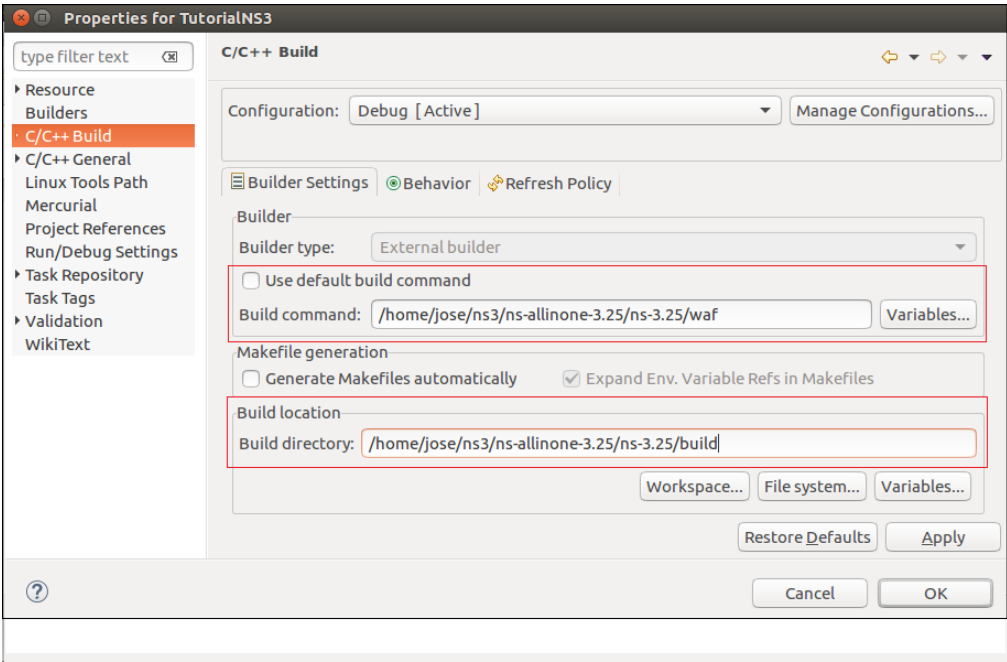


Figure 35. WAF Builder Settings.

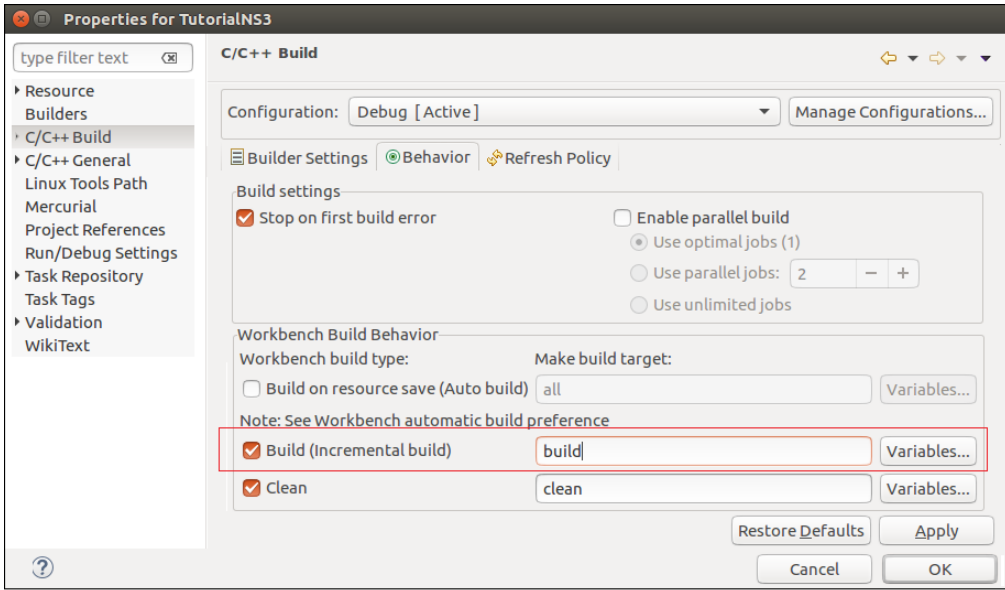


Figure 36. WAF Builder Settings.

* Click on the “Search Project” and type “scratch-simulator”

Switch to Environment tab, and click on “New” to create a new variable. Write “LD_LIBRARY_PATH” in the name field. On the other hand, in the value name file write the directory where build exists, in my case “/home/jose/ns3/ns-allinone-3.25/ns-3.25/build”. Finally, verify that "Append environment to native envir press the **“Apply”** and **“Close”** buttons.

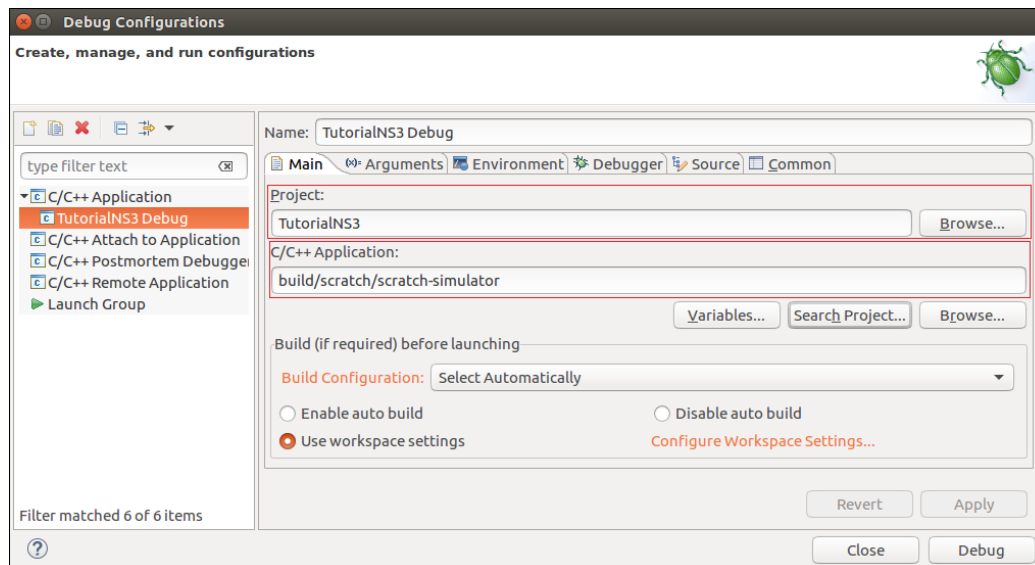


Figure 37. Debug configurations.

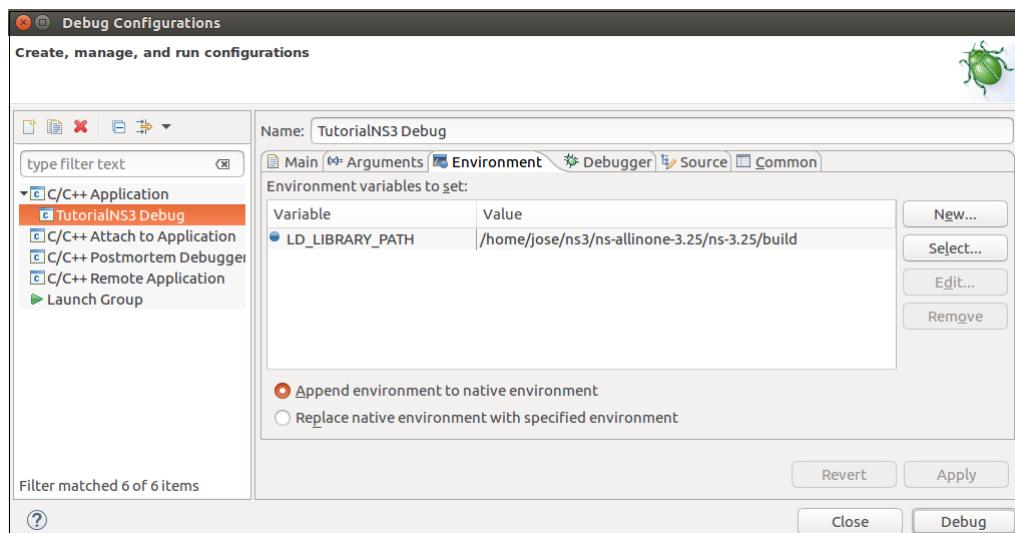


Figure 38. Debug configurations.

4.2.5. Configure External Runner

The purpose of this configuration is to link the commands used to run a program (See Fig. 11) to the Play button of Eclipse IDE interface. For this, follow the steps provided:

- * Go to Run → **External Tools** → **External Tools Configuration**
- * Add a new program (e.g. Run)
- * In location textbox, write your waf location (e.g. /home/jose/ns3/ns-allinone-3.25/ns-3.25/waf)
- * In Working Directory textbox, write your NS-3 directory (e.g. /home/jose/ns3/ns-allinone-3.25/ns-3.25)
- * In Argumentes textbox, type: -run "\$string_prompt" -vis
- * Click on **"Apply"**

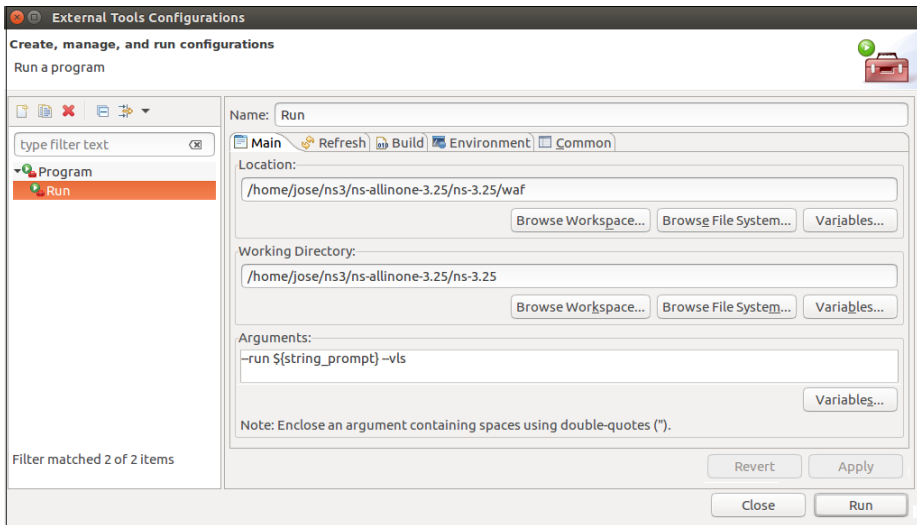


Figure 39. External Tools Configurations.

4.2.6. Running the first script in NS-3 on Eclipse IDE

After all installation steps have been completed, an example program must be run to verify the successful integration of NS-3 on Eclipse. It is worth mentioning that any NS3 script can be chosen. For this tutorial, the “ third.cc ” script of the default examples of NS-3 repository is shown. Then, the steps for running a program file:

- * Go to C++ Project_name (e.g. TutorialNS3) → **Examples** → **Tutorial** and double click on “third.cc” to open it.

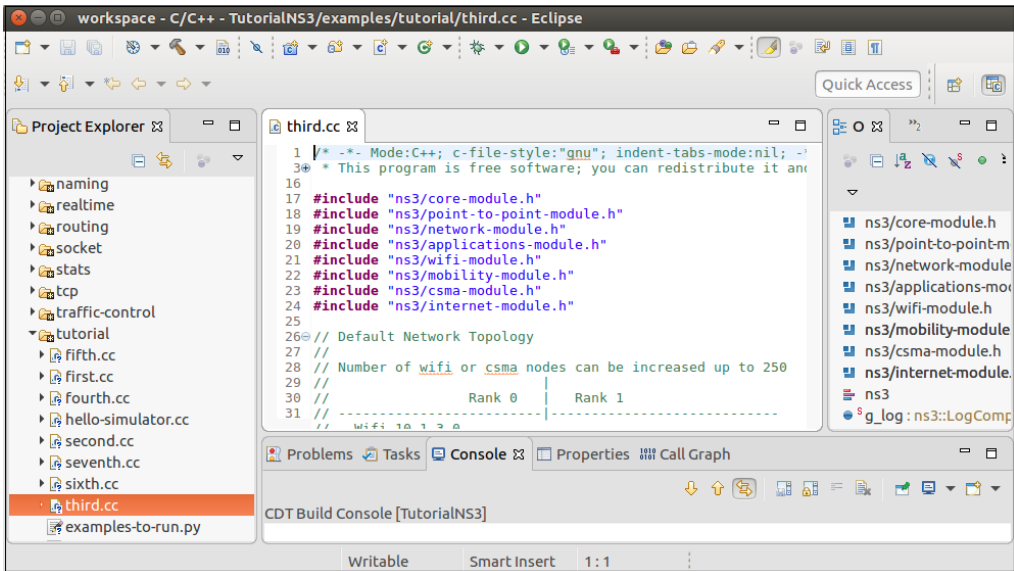


Figure 40. Example Script file.

From the Run menu, select the name of external runner created, in my case “Run”. The following window appears, write the name of the program file without using the extension.

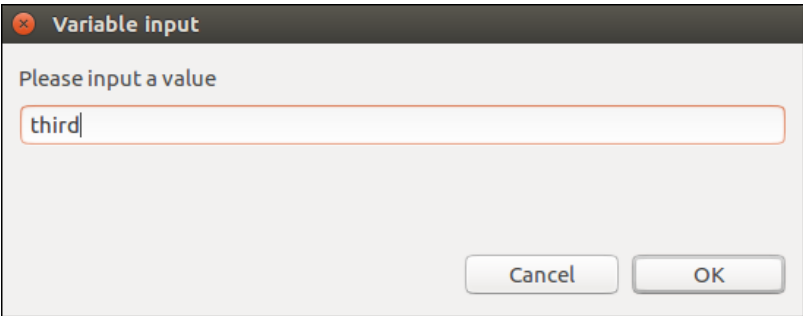


Figure 41. Enter the Project's name.

Finally, press “F3” button to simulate the program file

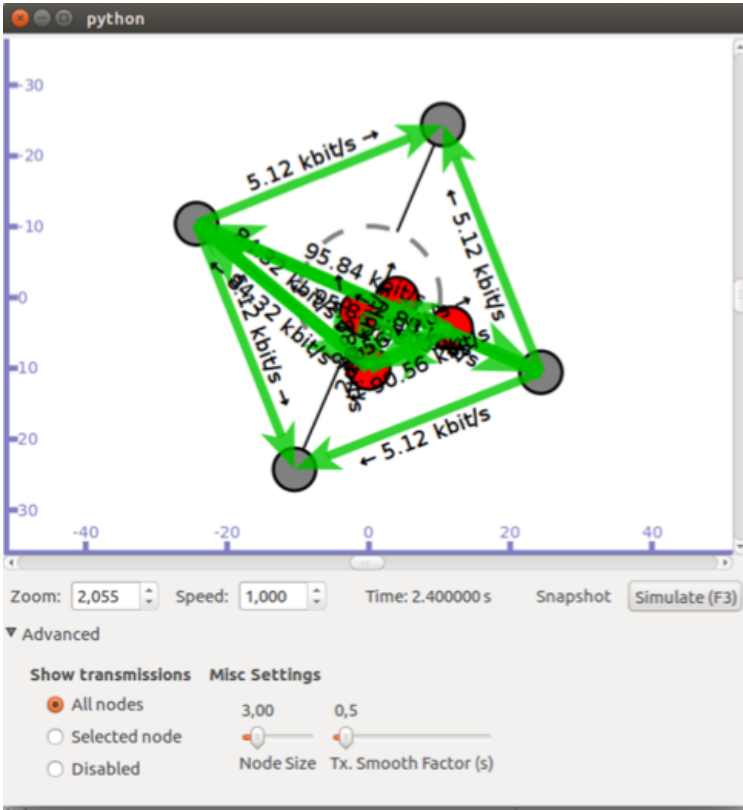


Figure 42. Output simulation.

This concludes our tutorial. To learn more, please read the references.

5. Additional Material

This tutorial includes a virtual machine with NS-3 simulator and Eclipse IDE.

6. Conclusions

A simulator is a very useful tool for learning and research. However, the first steps can be somewhat difficult and its complete domain is a long process. This work aims to ease the path of researchers by presenting a tutorial that indicates step by step how to install NS-3 in conjunction with Eclipse. Eclipse can become essential to debug elements (protocols) or create new ones. It help to track errors

and better understand the interactions between these components. As a future work we plan to develop a similar tutorial for another well-known simulator, OMNET ++. In addition, we want to develop a tutorial that facilitates the execution of block simulations and their subsequent analysis.

Bibliography

1. J.-D. Medjo, T. Kunz, and M. St-Hilaire, "Directional antennas in fanets: A performance analysis of routing protocols," *Selected Topics in Mobile and Wireless Networking (MoWNeT)*, 2017.
2. S. Siraj, A. Gupta, and R. Badgular, "Network simulation tools survey," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, no. 4, pp. 199–206, 2012.
3. K. Ravi and N. Sarma, "Simulation analysis of multi-dimensional wsns using ns-3," *Region 10 Conference, TENCON 2017*, 2017.
4. L. Weiwei, W. Xichen, Y. Wenli, Zhang ; Lin, and P. Chao, "Coordinative simulation with sumo and ns3 for vehicular ad hoc networks," *Communications (APCC)*, 2016.
5. M. Tom, Y. Zhenhui, and M. Gabriel-Miro, "Real time emulation of an lte network using ns-3." IET, 2013.
6. A. Fahimeh, K. Mohsen, and M. Seyed, "Analysis of qos parameters for video traffic in homeplug av standard using ns-3," *Smart Grids Conference (SGC)*, 2017.
7. A. Nosiba and A. Hamid, "Performance evaluation of tcp congestion control mechanisms using ns-2," *Basic Sciences and Engineering Studies (SGCAC)*, 2016.
8. J. Anton, M. Nair, G. Bok-Min, and M. Ezra, "Performance monitoring of power line communication mac protocol in the presence of impulsive noise using network simulator-3," *Communications (MICC)*, 2017.
9. Q. Bui and S. Houcke, "Network simulator: Importance of an accurate model of the physical layer," *Advanced Technologies for Communications (ATC)*, 2010.
10. K. Sunil, R. Priya, R. Radhakrishnan, and T. Malay, "An ns3 implementation of physical layer based on 802.11 for utility maximization of wsn," *Computational Intelligence and Communication Networks (CICN)*, 2016.
11. G. Carneiro, "Ns-3: Network simulator 3," in *UTM Lab Meeting April*, vol. 20, 2010.
12. D. Geer, "Eclipse becomes the dominant java ide," *Computer*, vol. 38, no. 7, pp. 16–18, 2005.
13. MediaWiki. (2017) Ns-3 installation. [Online]. Available: <https://www.nsnam.org/wiki/Installation>
14. hmgaudecker. (2013) Introduction to waf. [Online]. Available: <http://hmgaudecker.github.io/econ-project-templates/waf.html>