2-Days Workshop
on
Simulation of TCP Variants using NS-3
jointly organized by
Poornima University, Jaipur & IIIT-Kota (Mentored by MNIT, Jaipur)

Rahul Hada
rahul.hada@poornima.edu.in

Ashutosh Kumar
ashutosh.kumar@poornima.edu.in
Ambassador Vs Mercedes
I have no special talents. I am only passionately curious.

- Albert Einstein
Introduction to Simulation
Why Simulation?
What is Emulation?
History of Network Simulation-3
NS-3 Basics
Key Features of Network Simulation-3
Comparison with other Simulator
Installation of NS3
Introduction to Simulation

Definition

It is the imitation of the operation of a real-world process or system over time.¹

Requirement for Simulation

**Model** - It represents key characteristics or behaviour/function of the selected physical or abstract system or process.

[Model <-> System]

**Operation** - Simulation represents the operation of the system/model over time.

¹ en.wikipedia.org/wiki/Simulation.
Example-1

Horse Simulator: World War-I
More Examples

Human-in-the-loop: Outer Space

Soldier Test
Why Simulation Works

...simulation works because of the realism that can be built into the models to be analyzed through simulation.

A. Alan B. Pritsker, Pritsker Corporation
Benefits of Simulation

List of few benefits...

- Minimize of Spending
- Reduce Risk
- It provides experiential learning
- Easily Repeatable
- Scalable
- Flexible
Emulation

Emulation Mimic something close enough so that it can be substituted to the real things.
History of Network Simulation

1989 Srinivasan Keshav –> **REAL**

1995-1997 Steve McCanne, Sally Floyd, Kevin Fall –> **NS-1**
- Written in C++, Tcl-based scripting for simulation scenario.

1996-1997 ”Refactoring” by Steve McCanne –> **NS-2**
- Replace Tcl with Object Tcl(OTcl) of MIT

2006 NSF team (Henderson, Riley, Floyd, Roy) and INRIA team (Dabbous, Lacage) –> **NS-3**
- a new C++ core with Python bindings
- 26 releases and more than 175 open source contributors to date
NS-3 Basics

- An open source discrete event simulator
  - **Event model** packet transmission, receipt, timers etc.
  - Future events maintained in **sorted Event List**
  - Processing events results in **zero or more new events**

- Written in C++
  - Extensive use of **Templates**, **Smart Pointers**, **Callbacks**
  - C++ namespace (ns3)

- Simulation programs are C++ executables

- Python is used to bind public APIs provided

- NS-3 is **built as a library** which may be **linked to a C++ main program** defines the simulation topology and start the simulation.
Use of Library

- NS3 Library
  - Linked

- Building Topology & Start Simulation
  - MySim.cc
  - Main Program
Key Features of NS3

- **C++ and Python emphasis**
  - Many simulators use domain-specific modeling language to describe models and program flow.

- **Callback-driven events and connections**
  - Simulation events in ns3 are function calls that are scheduled to execute at a prescribed simulation time.

- **Flexible core with helper layer**
  - It provides "helper" layer APIs that provide easier-to-use functions with reasonable default behavior.

- **Emphasis on Emulation**
  - NS-3 packet objects are stored internally as packet byte buffers (similar to packets in real operating systems) ready to be serialized and sent to a real network interface.
**Key Features of NS3 (Cont...)**

- Numerous trace points enabled via callbacks
- Trace output in ascii, or pcap format
- Doxygen Documentation
- Use your favourite IDE
Installation of NS-3

Steps for Installation (Linux Platform)

- Step-1 Download source
- Step-2 Build from source
- Step-3 Testing
- Step-4 Run Example
Prerequisite Packages for Ubuntu

- **Minimal Requirements**
  - gcc, g++, python, python-dev, qt4-dev-tools, libxml2, libxml2-dev
- **Debugging and GNU Scientific Library (GSL) support**
  - gdb, valgrind, gsl-bin, libgsl0-dev, libgsl01dbl
- **Network Simulation Cradle**
  - flex, bison
- **Reading pcap packet traces**
  - tcpdump, wireshark
- **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
Prerequisite

```bash
sudo apt-get install gcc g++ python python-dev gdb valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev vtun lxc doxygen graphviz imagemagick texlive texlive-latex-extra texlive-generic-extra texlive-generic-recommended texinfo dia texlive-extra-utils texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev mercurial bzr libboost-all-dev
```
Using Tarball

- Download the current release of ns-3.25 from following link
  - http://www.nsnam.org/ns-3-25/
Building using `build.py`

- `build.py` program is used
- Goto directory `ns-allinone/` and type command
  - `./build.py -enable-examples -enable-tests`
  
  *(By default test and examples are not set to build)*

Building using `waf`

- `./waf` program is used
- Configure your project to build debug version
  - `./waf clean`
  - `./waf -d debug -enable-example -enable-tests configure`
Open Terminal

[Image of a terminal window with a search bar open to "ter" and applications such as Terminator, Terminal, UXTerm, and XTerm visible.]
GOTO ns-allinone-3.XX

tiger@tiger-rr:~$ cd workplace/
tiger@tiger-rr:~/workplace$ cd ns-allinone-3.20/
tiger@tiger-rr:~/workplace/ns-allinone-3.20$
BUILD from SOURCE

/build.py - -enable-examples - -enable-tests
Testing NS3

Run Script

cd ns3.XX/
./test.py -c core

Output
92 of 92 tests passed (92 passed, 0 failed, 0 crashed, 0 valgrind errors)
Running Hello Script

Hello Simulator

- Goto ns-3.XX folder
  
  ./waf \-run hello\-simulator

Scratch Folder Location

- Create-Modify-Copy simulator script
  
  .../ns-3allinone.XXX/ns3.XX/scratch/

- Now Run your first script
  
  ./waf \-run scratch/yourscriptname
Hello Simulator

```
tiger@tiger-rr:~/workplace/ns-allinone-3.20/ns-3.20
$ cd ns-allinone-3.20/
tiger@tiger-rr:~/workplace/ns-allinone-3.20
$ ls

Hello Simulator
```

Run First Sample Script

```
tiger@tiger-rr:~/workplace/ns-allinone-3.20/ns-3.20
$ cp examples/tutorial/first.cc scratch/```
Output of First Sample Script

```
tiger@tiger-rr:~$ ./waf --run scratch/first
Waf: Entering directory `/home/tiger/workplace/ns-allinone-3.20/ns-3.20/build'
Waf: Leaving directory `/home/tiger/workplace/ns-allinone-3.20/ns-3.20/build'
'build' finished successfully (2.146s)
At time 1s client sent 1024 bytes to 10.1.1.2 port 9
At time 1.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time 1.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time 1.00737s client received 1024 bytes from 10.1.1.2 port 9
```

NS3-Development Environment

Read and Modify

- **IDE**
  - Eclipse CDT

Easy Reference of Code

- **Doxygen**
  - Documentation System for C,C++,Java etc.
**Configure NS3 with Eclipse IDE**

- Download Linux version of Eclipse IDE for C/C++ Developer

- Configuration Steps as follows:-
  1. Install Eclipse
  2. Create new empty C++ project
  3. Import NS3 source code
  4. Configure Mercurial
  5. Configure Waf Builder
  6. Configure Debugger
  7. Run NS3 from Eclipse
1 Install Eclipse

- Synaptic Package Manager (*their are many ways of installation*)

2 https://www.nsnam.org/wiki/HOWTO\`configure\`Eclipse\`with\`ns-3.
Configure NS3 with Eclipse IDE

2 Create new empty C++ project

- Open Eclipse
  - Goto Help → Install new Software
  - Type http://cbes.javaforge.com/update
  - Click Add
  - Select Mercurial Eclipse
  - Click Next
  - Click Finish

- Create new C++ Project and with your preferred name

- Choose ns3 source path
  /home/openlab/ns-allinone-3.XX/ns-3.XX

- Right Click on Project → Team share project

- Click Next

- Select local Mercurial
3 Install Mercurial Plug-in

Work with:

```
http://cbes.javaforge.com/update
http://cbes.javaforge.com/update
```

- Codebeamer Eclipse Studio (with Mylyn)
- MercurialEclipse

MercurialEclipse is a plugin providing support for the highly popular Mercurial Distributed Version Control System within Eclipse.

- Show only the latest versions of available software
- Show items by category
- Show only software applicable to target environment
- Contact all update sites during install to find required software
4 Import NS3 source code

[Image of Eclipse IDE configuration for NS3 project]

Project name: NS3
Location: /home/tiger/workplace/ns-allinone-3.20/ns-3.20

Cross GCC
Linux GCC
5 Configure Waf Builder

- Right click on project and click on **Properties**. Then click on **C/C++ build**. After that unchecked the boxes:-
  - Use default build command
  - Generate make file automatically
- Insert this path as **build command**
  - `/home/xxxx/ns-allinone-3.xx/ns.xx/waf`
- Insert this path as **build directory**
  - `/home/xxxx/ns-allinone-3.xx/ns.xx/build`
- Click in Behavior tab and remove the all option in front of the Build (Incremental build) checkbox and then write build
6 Configure Debugger

- Go to **Run -> Debug Configuration** and create a **New configuration**
- Click on Browse button under the project section and select your project
- Click on the Search Project button under the C/C++ Application section and type scratch-simulator
- Go to **Environment Tab** and click on New button to create a new variable. You will need these
  - **LD_LIBRARY_PATH**
  - `/home/xxxx/ns-allinone3.xx/ns-3.xx/build`
- Be sure the “Append Environment to native environment” is selected
sudo apt-get install doxywizard

Step-1 → Select Working Directory
Step-2 → Configure Doxygen
  - Project
  - Mode
  - Output
  - Diagrams
Install NetAnim

- **Prerequisites**
  - Mercurial
    - `sudo apt-get install mercurial`
  - QT4 development packages
    - `sudo apt-get install qt4-dev-tools`
- **Installation**: Goto `ns-allinone3.xx` folder -> open `netanim` folder
  - make clean
  - qmake NetAnim.pro
  - make