ns-3 training

Tom Henderson

ns-3 annual meeting 2019

June 17-21, Florence, Italy

UNIVERSITY OF WASHINGTON
Next steps

> Code organization and build system
> Documentation system
> Packet objects and queues
> Walkthrough of ‘mm1-queue.cc’ example
  – Simple experiment management
  – Objects, attributes, tracing
  – Logging and debugging
Software orientation

Key differences from other network simulators:

1) Command-line, Unix orientation
   – vs. Integrated Development Environment (IDE)

2) Simulations and models written directly in C++ and Python
   – vs. a domain-specific simulation language
Submodule vectors, gate vectors and multiple connections are illustrated in the following example:

```c
simple Hub
gates:
  out: outport[];
endsimple

simple Station //...

module Star
  submodules:
    hub: Hub
      gatesizes: outport[4];
      station: Station[4];
      connections:
        for i=0..3 do
          hub.outport[i] --> station[i].in;
        endfor
  endmodule
```

The result of the above is depicted in Fig.4.

Example of OMNeT++ Network Description (NED) language
Figure excerpted from [http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm](http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm)
ns-3 does not have a graphical IDE

Figure source: https://www.comsol.com/comsol-multiphysics

ns-3 Training, June 2019
ns-3 uses outside programs for graphics

LTE radio environment map (REM)

Will experiment with this on Tuesday
ns-3 users typically write scripts to plot

Animated

EmsVideo_1_Server 942.36392953 RX 1012 1061 U 2395
EmsVideo_1_Client 942.37317727 TX 1012 1061 U 2398
EmsVideo_1_Client 942.377 RX 64 113 U 2397
WebBrowsingGraphics_0_Server 942.38092876 TX 1024 1073 U 2399
WebBrowsingGraphics_0_Client 942.394 RX 1024 1073 U 2399
AvlAssetPerimeter_1_Server 942.42492988 RX 1408 1457 U 2401

Used to measure KPIs

PTT Floor granted Release
(KPI 1) Access time
Talk spurt
Talk spurt
Sender Receiver
(KPI 3) Mouth to ear latency

Throughput vs. time for incident scenario

Network traffic throughput vs time

ns-3 Training, June 2019
Visualization

• No preferred visualizer for ns-3
• Two tools have been developed over the years, with some scope limitations
  – Pyviz
    • FlowMonitor (statistics with Pyviz linkage)
  – NetAnim (George Riley and John Abraham)

• Support is lagging for these tools (help wanted)
PyViz overview

• Developed by Gustavo Carneiro
• Live simulation visualizer (no trace files)
• Useful for debugging
  – mobility model behavior
  – where are packets being dropped?
• Built-in interactive Python console to debug the state of running objects
• Works with Python and C++ programs
Pyviz and FlowMonitor

• Example screenshot from:

```bash
./waf --run src/flow-monitor/examples/wifi-olsr-flowmon.py --vis
```
Enabling PyViz in your simulations

• Make sure PyViz is enabled in the build

```
<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLite stats data output</td>
<td>not enabled (library 'sqlite3' not found)</td>
</tr>
<tr>
<td>Tap Bridge</td>
<td>enabled</td>
</tr>
<tr>
<td>PyViz visualizer</td>
<td>enabled</td>
</tr>
<tr>
<td>Use sudo to set suid bit</td>
<td>not enabled (option --enable-sudo not selected)</td>
</tr>
</tbody>
</table>
```

• If program supports CommandLine parsing, pass the option

   ```
   --SimulatorImplementationType=
   ns3::VisualSimulatorImpl
   ```

• Alternatively, pass the "--vis" option
FlowMonitor

• Network monitoring framework found in src/flow-monitor/

• Goals:
  – detect all flows passing through network
  – stores metrics for analysis such as bitrates, duration, delays, packet sizes, packet loss ratios

Plan to discuss more on Tuesday

NetAnim

- "NetAnim" by George Riley and John Abraham
NetAnim key features

• Animate packets over wired-links and wireless-links
  – limited support for LTE traces
• Packet timeline with regex filter on packet meta-data.
• Node position statistics with node trajectory plotting (path of a mobile node).
• Print brief packet-meta data on packets
Software organization

• Two levels of ns-3 software and libraries

1) Several supporting libraries, not system-installed, can be in parallel to ns-3

Netanim  pybindgen  Click routing  ● ● ●  ns-3

2) ns-3 modules exist within the ns-3 directory

module  module  module  ● ● ●  module  module  module

ns-3 Training, June 2019
Typical module source code organization

model/
examples/
test/
bindings/
doc/
wscript/
Modules in ns-3 mainline

- devices
  - bridge
  - csma
  - emu
  - point-to-point

- protocols
  - internet-apps
  - aodv
  - dsdv
  - olsr

- network
  - Packets
  - Packet Tags
  - Packet Headers
  - Pcap/ascii file writing

- core
  - network
  - time-arithmetic
  - events
  - scheduler

- Utilities
  - visualizer
  - config-store
  - flow-monitor
  - netanim
  - stats
  - topology-reader

- Ir-wpan
- wifi
- wimax

- Node class
  - NetDevice ABC
  - Address types (IPv4, MAC, etc.)
  - Queues
  - Socket ABC
  - IPv4 ABCs
  - Packet sockets

- Smart pointers
- Dynamic types
- Attributes
- Callbacks
- Tracing
- Logging
- Random Variables

- openflow
- click
- mix-vector-routing

- mpi
- energy
- mobility
- propagation
- config
- store
- netanim

- BRITE
ns-3 programs

- ns-3 programs are C++ executables that link the needed shared libraries
  - or Python programs that import the needed modules
- The ns-3 build tool, called 'waf', can be used to run programs
- waf will place headers, object files, libraries, and executables in a 'build' directory
Python bindings

- ns-3 uses a program called PyBindGen to generate Python bindings for all libraries

---

Diagram:

1. **C++ header**
2. **Intermediate Python program**
3. **C++ bindings code**
4. **Python module**

- **CastXML (py)gccxml**
- **PyBindGen**
- **C++ compiler**
Python bindings status

- API scanning for Python used to use a tool called gccxml
- ns-3 has moved to the successor, CastXML
  - requires a development installation of clang
- Automated testing currently only for Linux 64-bit systems
  - MacOS API scanning is not tested
waf operation

• This slide is a placeholder to demonstrate Waf operation
  – ‘waf build’ will compile and link source code into executables
  – ‘waf --run’ will run an executable in a special shell that knows the path to ns-3 libraries
  – New option: ‘waf --run-no-build’ will skip the build step
waf configuration

- Key waf configuration examples
  - ./waf configure
    - --enable-examples
    - --enable-tests
    - --disable-python
    - --enable-modules

- Whenever build scripts change, need to reconfigure

Demo: ./waf --help
  ./waf configure --enable-examples --enable-tests --enable-modules='core'

Look at: build/c4che/_cache.py
def build(bld):
    obj = bld.create_ns3_module('csma', ['network', 'applications'])
    obj.source = ['model/backoff.cc',
                  'model/csma-net-device.cc',
                  'model/csma-channel.cc',
                  'helper/csma-helper.cc',
                 ]
    headers = bld.new_task_gen(features=['ns3header'])
    headers.module = 'csma'
    headers.source = ['model/backoff.h',
                      'model/csma-net-device.h',
                      'model/csma-channel.h',
                      'helper/csma-helper.h',
                     ]

    if bld.env['ENABLE_EXAMPLES']:
        bld.add_subdirs('examples')

    bld.ns3_python_bindings()
waf build

• Once project is configured, can build via
  ./waf build or ./waf

• waf will build in parallel on multiple cores

• waf displays modules built at end of build

Demo: ./waf build

Look at: build/ libraries and executables
Running programs

• ./waf shell provides a special shell for running programs
  – Sets key environment variables

  ./waf --run sample-simulator
  ./waf --pyrun src/core/examples/sample-simulator.py
Build variations

- Configuring a build type is done at waf configuration time
- debug build (default): all asserts and debugging code enabled
  ./waf -d debug configure
- optimized
  ./waf -d optimized configure
- static libraries
  ./waf --enable-static configure
Controlling the modular build

• One way to disable modules:
  – ./waf configure --enable-modules='a','b','c'

• The .ns3rc file (found in utils/ directory) can be used to control the modules built

• Precedence in controlling build
  1) command line arguments
  2) .ns3rc in ns-3 top level directory
  3) .ns3rc in user's home directory

Demo how .ns3rc works
Building without wscript

- The scratch/ directory can be used to build programs without wscripts

Demo how programs can be built without wscripts
Integrating other tools and libraries
Other libraries

• more sophisticated scenarios and models typically leverage other libraries

• ns-3 main distribution uses optional libraries (libxml2, gsl, mysql) but care is taken to avoid strict build dependencies
  – The Waf wscripts can be consulted as examples
  – example: sqlite3 in src/stats/wscript

• the 'bake' tool (described later) helps to manage library dependencies

• users are free to write their own Makefiles or wscripts to do something special
CORE emulator

The Common Open Research Emulator (CORE) is a tool for emulating networks on one or more machines. You can connect these emulated networks to live networks. CORE consists of a GUI for drawing topologies of lightweight virtual machines, and Python modules for scripting network emulation.
mininet emulator

---

**GitHub**

mininet / mininet

**Home**  **Pages**  **History**

**Link modeling using ns 3**

**Contents**

- Introduction
  - ns-3 emulation features
  - Link simulation with ns-3
- Details
  - How to achieve communication of ns-3 process with TAP interfaces in distinct namespaces?
  - Architecture: single ns-3 thread or multiple processes?
- Code
  - Mininet
  - ns-3 patches

---

ns-3 Training, June 2019
Co-simulation frameworks have emerged

• PNNL's FNCS framework integrates ns-3 with transmission and distribution simulators

ns-3 App Store

- Project is migrating away from a centralized repository to a modular system called the ‘ns-3 App Store’
  - https://apps.nsnam.org
Documentation overview

- Placeholder slide: online browsing of
  - Doxygen
  - ns-3 manual, model library, tutorial
  - wiki
  - command-line help