ns-3 development overview
ns-3 tutorial agenda

• 3:00-4:30: ns-3 current capabilities
  • Project overview
  • Walkthrough of basic simulation scenario
  • Parallel simulations and visualization
  • Emulation

• 4:30-4:40: 10-minute break

• 4:40-5:45: Work in progress
  • ns-3 development process
  • Automation
  • Direct code execution
  • Virtual machine and testbed integration

• 5:45-6:00: Q & A
ns-3 development process

ns-3 is run as an open source project backed by research funding

- GPLv2 licensing stance
- open mailing lists
- use standard tools (Mercurial, Bugzilla, Mediawiki, GNU/Linux development)
- 13 maintainers worldwide
**ns-3 development process**

- date-driven quarterly releases

![Diagram of the ns-3 development process]

- All code for merge to ns-3 is openly reviewed by maintainers
  - Syntactic (style) reviews
  - Design reviews
  - Documentation and tests
current merge queue

• ns-3.10 release (January 2010)
  – new TCP model with modular congestion control
  – Virtual Access Point (VAP) for WiFi
  – BulkSend application
  – Pyviz visualizer
  – Energy model for WiFi
  – DSDV routing for IPv4
  – PhySim for WiFi
ns-3 and Google Summer of Code

Click Modular Router
Lalith Suresh

ns-3 OpenFlow
Blake Hurd

Underwater Acoustic Networking
Andrea Sacco

ns-3 LTE
Giuseppe Piro
Other announced projects

- Wireless jamming model
- MPLS
- VANET mobility model
- TDMA
- TCP Vegas
- DSR routing
- SimpleWireless model
- Zigbee, WPAN, and 6LOWPAN
- Chord/DHash DHT
- Synchronized emulation
- 802.11n
- TMix and DelayBox
- ns-3-simu
- multi-core parallelization
Modularity and model store

- Moving to a modular build and package management system
  - ns-3 project maintains the core
  - models may be enabled/disabled
  - other research groups may separately maintain their own models
    - maintainers can still provide reviews
    - common package metadata format used to inform ns-3 build system
automation
Overview of ns-3 features

Start with a research question

Models:
- WiFi intro
- TCP

Real-time scheduler
Emulation modes
Debugging

Visualization

Output Analysis

Examples

Topology Definition

Models

Configuration

Execution

Modify scenario, or perform independent replication

Helper APIs and containers

Attributes
Names
Command line args
Default values
Env. variables

Tracing
Wireshark
Statistics framework
Random variables
Motivation

Network simulation is no easy business. One must:

• Build a model that is consistent.
• Describe the simulation model for a given simulator.
• Design and execute experiments.
• Process output data using sound methodologies.

We raise the level of abstraction on the user interface to the network simulator to support the needs of both experts and novices.

We can address issues that undermine credibility.
Frameworks for ns-3

- NSF CISE Community Research Infrastructure
  - University of Washington (Tom Henderson), Georgia Tech (George Riley), Bucknell Univ. (Felipe Perrone)
  - Project timeline: 2010-14
Automation

• Task led by Felipe Perrone, Bucknell Univ.
• Inspired by SWAN-Tools and ANSWER frameworks.
• User interfaces, description languages, and tools for automation of experiments.
• Model composition, structural validation, control of experiments, data processing and storage, and archiving experimental setup.
Topology generation

• Integrate BRITE topology generator (Boston Univ.) into framework.
• BRITE is downloaded into distribution and compiled by the ns-3 build system.
• The ns-3 simulation script uses a topology helper which reads a BRITE configuration file, receives results from BRITE, and builds the ns-3 topology.
Broader use case

• Provide a model and a description of experiment.
• Framework generates design of experiment space, distribute simulation runs to machines, collect results and archive in persistent storage.
• User mines storage to find, extract, and visualize results.
General architecture
Direct Code Execution
Virtual machines and ns-3
Goals

• Lightweight virtualization of kernel and application processes, interconnected by simulated networks

• Benefits:
  – Implementation realism in controlled topologies or wireless environments
  – Model availability

• Limitations:
  – Not as scalable as pure simulation
  – Runs in real-time
  – Integration of the two environments
**netns3**

- Written by Tom Goff (Boeing)
  - Documentation and prototype posted on wiki
- Basic Python-based framework using ns-3 Python bindings, RPyC distributed computing library, and ns-3 tap bridge framework

Combining Linux namespaces and ns-3 can provide a framework for network emulation. Native code can then run in real-time and produce and/or consume "live" network traffic. Linux network namespaces (netns) are used as virtual hosts and a virtual network topology is created from ns-3 models running the real-time scheduler.

This page describes one way to use ns-3 for Linux-based network emulation on an Ubuntu 9.10 system. This is similar to the approach described in HOWTO Use Linux Containers to set up virtual networks. The main distinctions are:

- Less isolation between virtual nodes
- Control groups are not used
- The host filesystem is shared by default
- Network devices are directly assigned to virtual nodes without using an intermediate bridge

Python is used to create and configure Linux namespaces and ns-3 objects. Python glue also serves as the interface between netns and ns-3 by managing information used by both (e.g., IP addresses).

The sections below give a quick overview of how to setup and run some basic network emulation scenarios.
netns3 demo

```python
from ns3 import NetDevice, Node

node = Node()
iface = node.AddInterface()
iface.SetAddress(ns3.Ipv4Address('10.0.0.1'))
```
CORE is an open source project

- Web site and code repository hosted by NRL ITD

- Open source licensed
  - modified BSD license

- Source code at NRL SVN

- Wiki/Bug tracker:
  - http://code.google.com/p/coreemu/

- Mailing lists at NRL:
  - core-users
  - core-dev
Future work: Integrating ns-3 and GUIs

• Example CORE and ns-3
  – CORE could glue virtual machines to ns-3 networks
Other recent related work

CORE is the Common Open Research Emulator that controls lightweight virtual machines and a network emulation subsystem (more on this later).

NEPI/NEF: Using Independent Simulators, Emulators, and Testbeds for Easy Experimentation
   – Lacage, Ferrari, Hansen, Turletti (Roads 2009 workshop)

EMANE is an Extendable Mobile Ad-hoc Network Emulator that allows heterogeneous network emulation using a pluggable MAC and PHY layer architecture.
   – http://labs.cengen.com/emane
   – being integrated with CORE
Scaling time in virtualized environments

• Synchronized Network Emulation - RWTH Aachen University
  – Modified Xen

• VAN Testbed – Telcordia/CERDEC
  – Modified Xen

• Linux Time namespace - Jeff Dike (UML creator)
  – Add a time namespace to the Linux kernel, allowing for gettimeofday() offsets
NEPI
Resources

Web site:
http://www.nsnam.org

Mailing list:
http://mailman.isi.edu/mailman/listinfo/ns-developers

IRC: #ns-3 at freenode.net

Tutorial:
http://www.nsnam.org/docs/tutorial/tutorial.html

Code server:
http://code.nsnam.org

Wiki:
http://www.nsnam.org/wiki/index.php/Main_Page