ns-3 Training

Session 1: Wednesday March 18

MNM Workshop
March 2015

ns-3 training goals

- Learn about the project scope, and where to get additional help
- Understand the architecture and design goals of the software
- Introduce how to write new code for the simulator
- Learn about selected topics in more detail
- Answer your questions

ns-3 training agenda

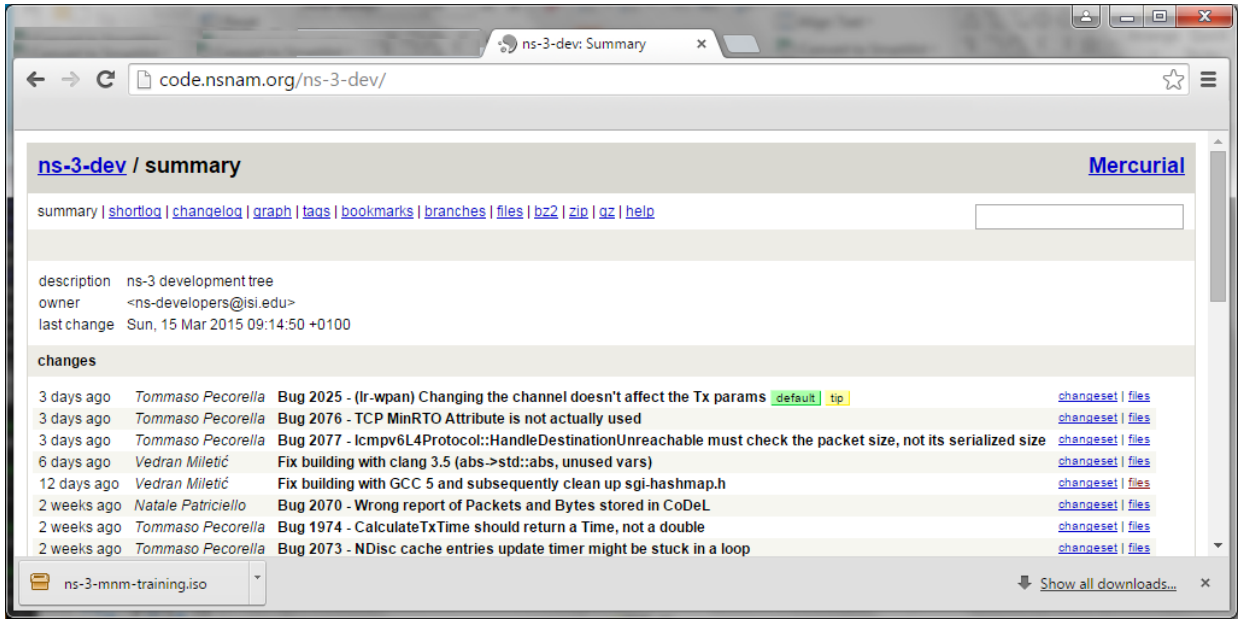
- Wednesday: Overview of ns-3
  - Session 1: Project overview
  - Session 2: Parallel, distributed simulations
  - Session 3: Survey of capabilities
- Thursday: Closer look at the code
  - Software core (architecture, object model, packets, scheduler, etc.)
  - Mobility
  - Tracing and output data
  - Wi-Fi and LTE
  - Writing and debugging code for ns-3

Options for working along

1) Download the required packages onto your (Linux, OS X, or BSD) system

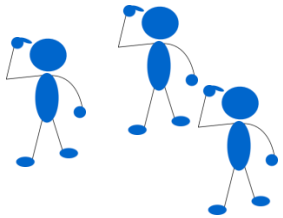
2) Download or copy the ISO image (Live DVD)

3) Browse the code online: https://code.nsnam.org

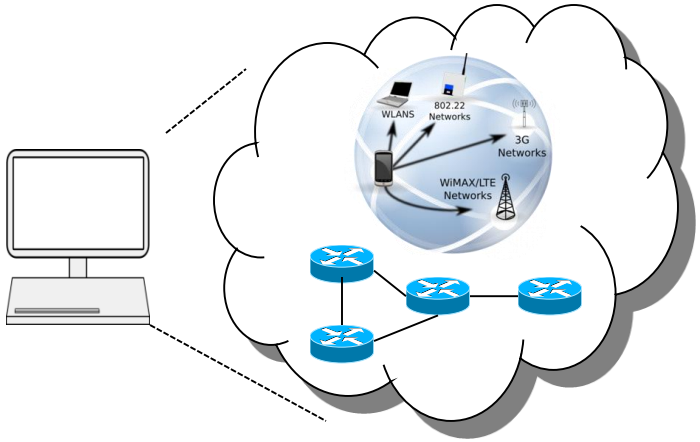


ns-3: An Open Source Network Simulator

- ns-3 is a ***discrete-event network simulator*** targeted for ***research and educational use***



model developers



ns-3 software

Research

NS-3 Consortium

ns-3 maintainers

Education

ns-3 project goals

Develop an extensible simulation environment for networking research

1) a tool aligned with the experimentation needs of modern networking research

2) a tool that elevates the technical rigor of network simulation practice

3) an open-source project that encourages community contribution, peer review, and long-term maintenance and validation of the software

How the project operates

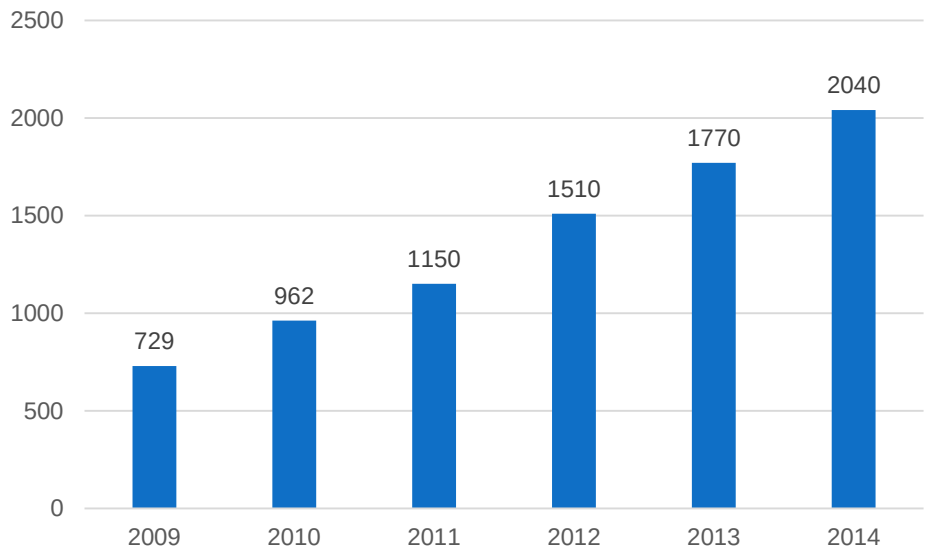
- Project provides three annual software releases
- Users interact on mailing lists and using Bugzilla bug tracker
- Code may be proposed for merge
  - Code reviews occur on a Google site
- Maintainers (one for each module) fix or delegate bugs, participate in reviews
- Project has been conducting annual workshop and developer meeting around SIMUTools through 2013
  - Some additional meetings on ad hoc basis
- Google Summer of Code (March-August) six of the past seven summers

Sustainment

- The NS-3 Consortium is a collection of organizations cooperating to support and develop the ns-3 software.
- It operates in support of the open source project
  - by providing a point of contact between industrial members and ns-3 developers,
  - by sponsoring events in support of ns-3 such as users' days and workshops,
  - by guaranteeing maintenance support for ns-3's core, and
  - by supporting administrative activities necessary to conduct a large open source project.

How many ns-3 publications?

- Google Scholar search of keyword 'ns-3 simulator'
  - Advanced search filters: English only, excluding patents and citations
- Results by year (searched early March 2015):



Acknowledgment of support

National Science Foundation
WHERE DISCOVERIES BEGIN

Naval Research Laboratory
Washington, DC

Inria informatics mathematics

PLANETE

Google Summer of Code 2012

University of Washington 1861

Georgia Tech

INESCPORTO
LABORATÓRIO ASSOCIADO

ISI
Information Sciences Institute

What is ns-3?

Software overview

- ns-3 is written in C++, with bindings available for Python
  - simulation programs are C++ executables or Python programs
  - ~350,000 lines of C++ (estimate based on cloc source code analysis)
- ns-3 is a GNU GPLv2-licensed project
- ns-3 is mainly supported for Linux, OS X, and FreeBSD
  - Windows Visual Studio port available
- ns-3 is not backwards-compatible with ns-2

Discrete-event simulation basics

- Simulation time moves in discrete jumps from event to event
- C++ functions schedule events to occur at specific simulation times
- A simulation scheduler orders the event execution
- Simulation::Run() executes a single-threaded event list
- Simulation stops at specific time or when events end

Software orientation

Key differences from other network simulators:

1) Command-line, Unix orientation

- vs. Integrated Development Environment (IDE)

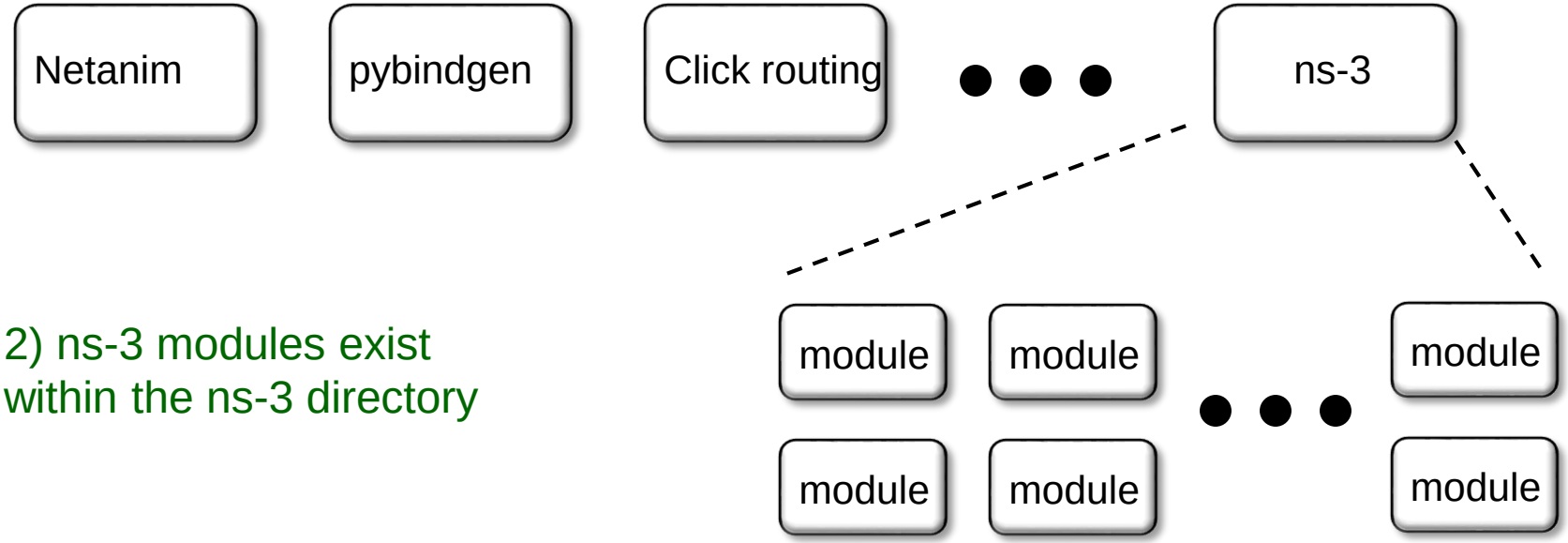
2) Simulations and models written directly in C++ and Python

- vs. a domain-specific simulation language

Software organization

- Two levels of ns-3 software and libraries

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



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

Current models

devices

- bridge
- mesh
- csma
- spectrum
- emu
- tap-bridge
- point-to-point
- uan
- lte
- virtual-net-device
- lr-wpan
- wifi
- wimax

- applications
- internet (IPv4/v6)
- sixlowpan
- wave
- network
- core

- energy
- mpi
- mobility
- propagation
- buildings

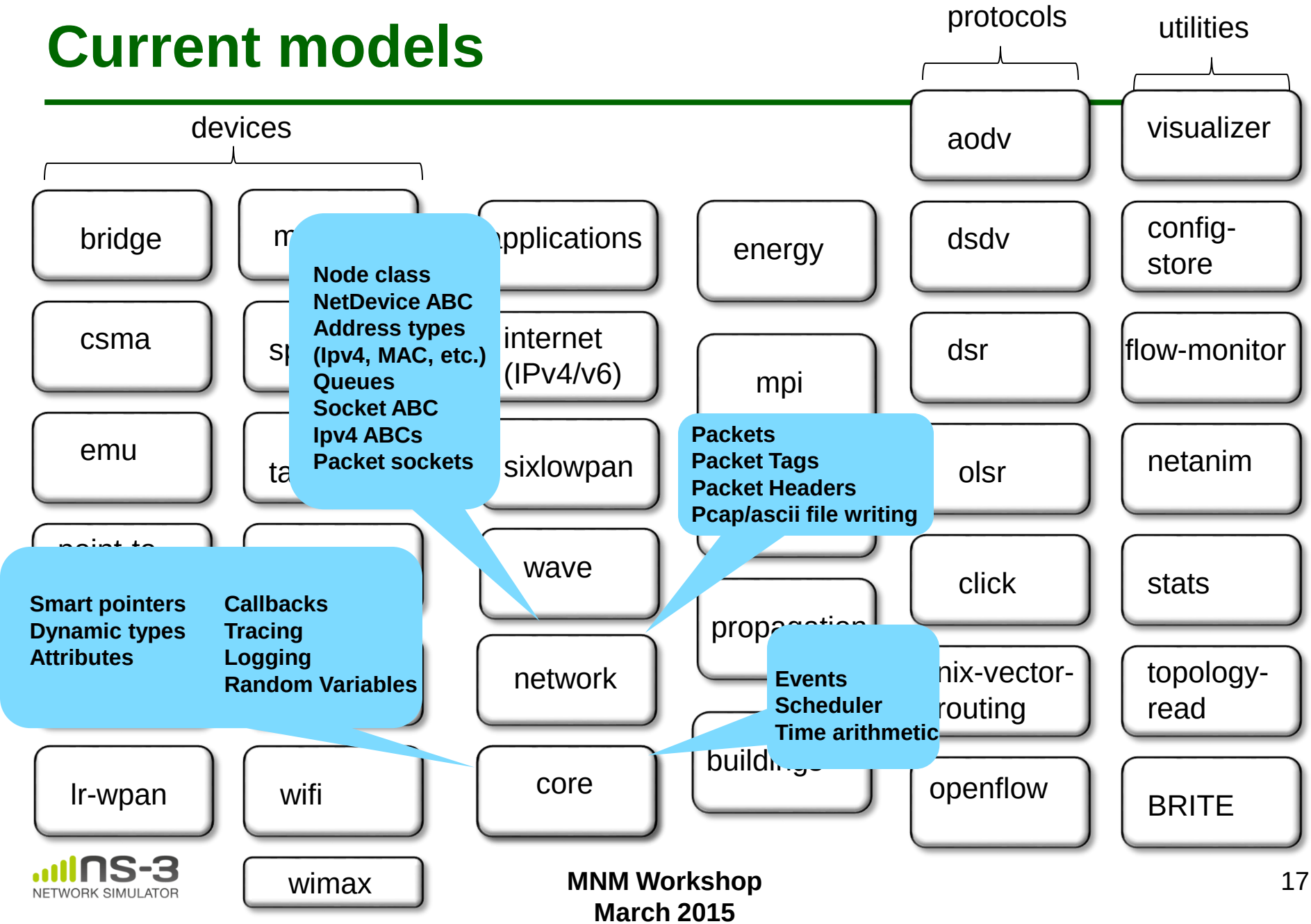
protocols

- aodv
- dsdv
- dsr
- olsr
- click
- nix-vector-routing
- openflow

utilities

- visualizer
- config-store
- flow-monitor
- netanim
- stats
- topology-read
- BRITE

Current models



Module organization

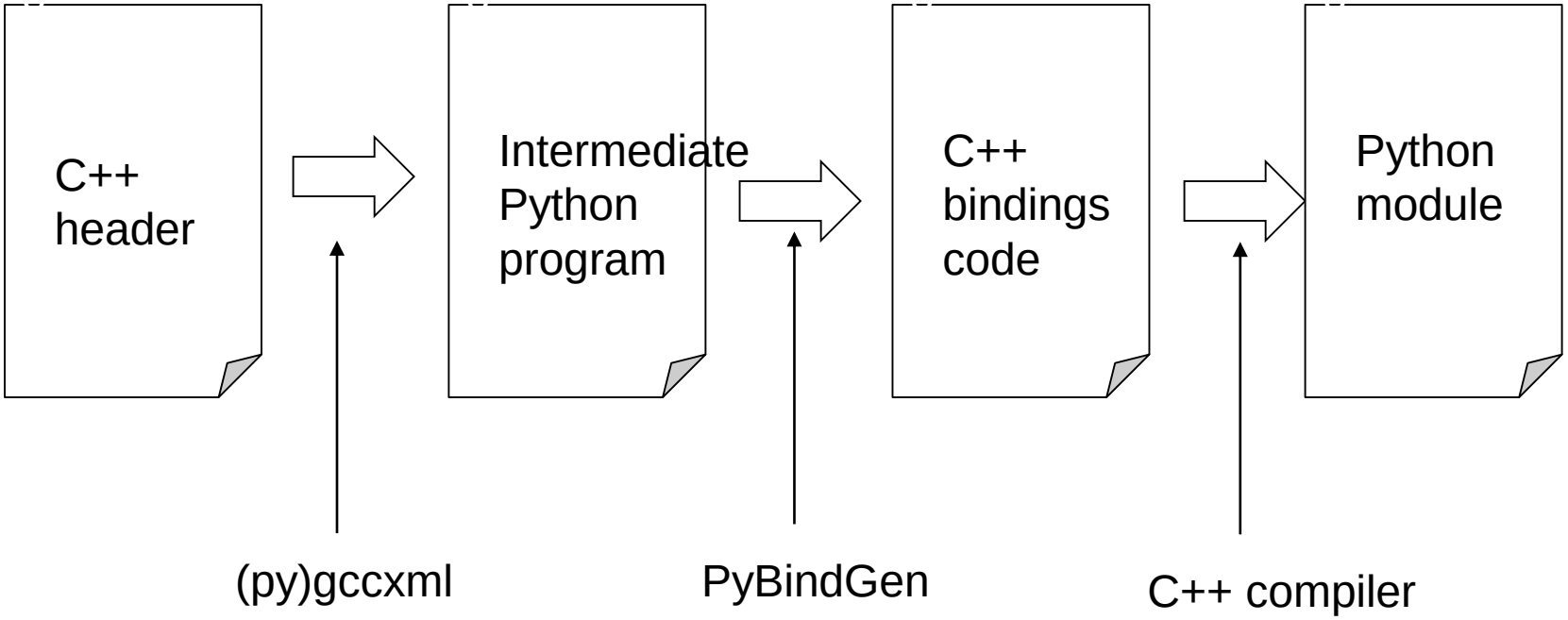
- models/
- examples/
- tests/
- bindings/
- doc/
- wscript

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



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 'bake' tool (described later) helps to manage library dependencies
- users are free to write their own Makefiles or wscripts to do something special

Gnuplot

- `src/tools/gnuplot.{cc,h}`
- C++ wrapper around gnuplot
- classes:
  - `Gnuplot`
  - `GnuplotDataset`
    - `Gnuplot2dDataset, Gnuplot2dFunction`
    - `Gnuplot3dDataset, Gnuplot3dFunction`

Enabling gnuplot for your code

- `examples/wireless/wifi-clear-channel-cmu.cc`

```
CommandLine cmd;
cmd.Parse (argc, argv);

Gnuplot gnuplot = Gnuplot ("clear-channel.eps");

for (uint32_t i = 0; i < modes.size (); i++)
  {
    std::cout << modes[i] << std::endl;
    Gnuplot2dDataset dataset (modes[i]);
```

produce a plot file that will generate an EPS figure

one dataset per mode

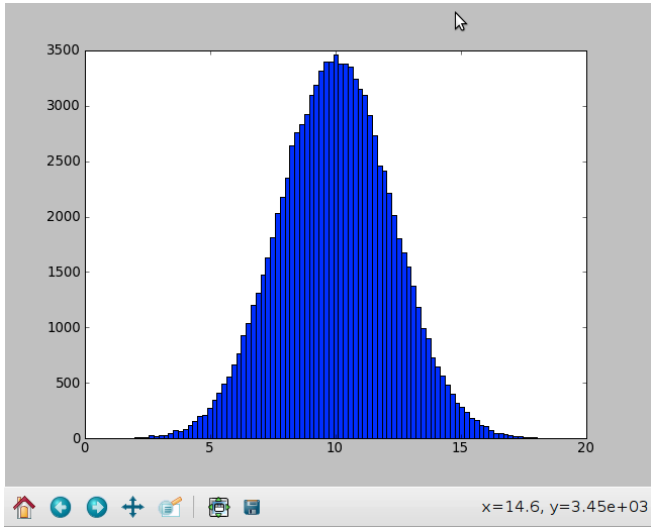
```
    uint32_t pktsRecvd = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
    dataset.Add (rss, pktsRecvd);
  }

gnuplot.AddDataset (dataset);
```

Add data to dataset

Add dataset to plot

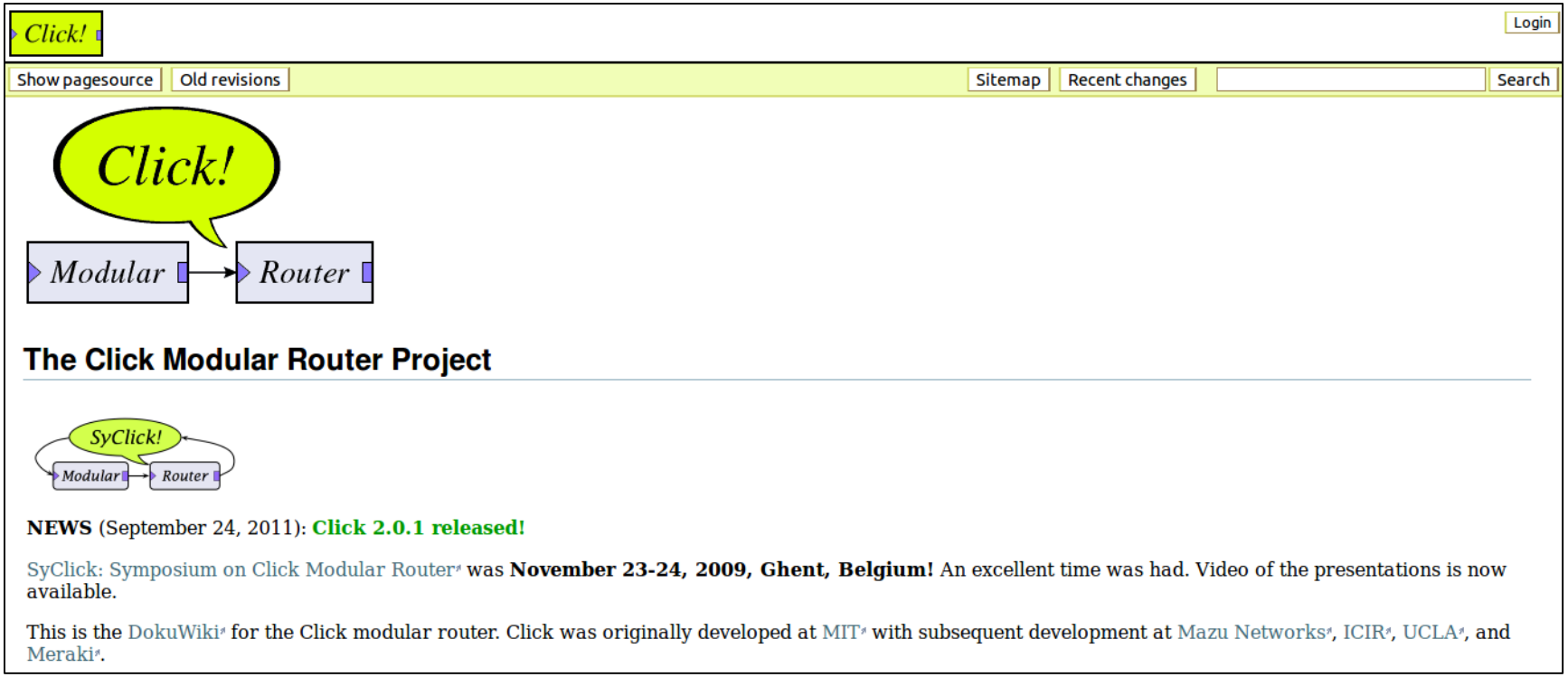
Matplotlib

- `src/core/examples/sample-rng-plot.py`




```
# Demonstrate use of ns-3 as a random number generator integrated
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial

import numpy as np
import matplotlib.pyplot as plt
import ns.core

# mu, var = 100, 225
rng = ns.core.NormalVariable(100.0, 225.0)
x = [rng.GetValue() for t in range(10000)]

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)

plt.title('ns-3 histogram')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

Click Modular Router



OpenFlow Switch

Please Note: This website has been archived and is no longer maintained.
See the Open Networking Foundation for current OpenFlow-related information.

OpenFlow

Google™ Custom Search | Search

Home | Videos | Documents | News | Research | About

Page | Discussion | View source | History

MPLS with OpenFlow/SDN

Contents [hide]

Motivation

MPLS networks have evolved over the last 10-15 years to become critically important for ISPs. They provide two key services: traffic engineering in IP networks and L2 or L3 enterprise VPNs. However as carriers deploy MPLS networks, they find that (a) even though the MPLS data plane was meant to be simple, vendors end up supporting MPLS as an additional feature on complex, energy hogging, expensive core routers; and (b) the IP/MPLS control plane has become exceedingly complex with a wide variety of protocols tightly intertwined with the associated data-plane mechanisms.

Quick Navigation

- » OpenFlow Specs
- » Bug Tracking
- » Wiki
- » Legal
- » Log in

OpenFlow White Paper

Download the OpenFlow Whitepaper (PDF)

OpenFlow Specification

Download v1.1.0 Implemented (PDF)

Wiki Tools

Personal tools

- » Log in

Navigation

- » OpenFlow.org

CORE emulator

Networks and Communication Systems Branch

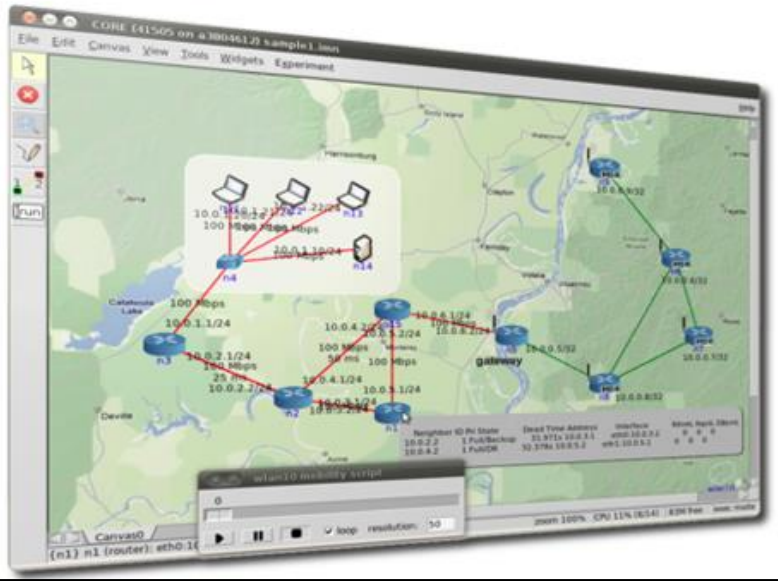
Focus Areas | Projects | Products | Organization

/ NRL / ITD / NCS / Common Open Research Emulator (CORE)

Common Open Research Emulator (CORE)

NCS Home
Focus Areas
Projects
Products
Organization

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 | This repository | Search or type a command | Explore | Features | Enterprise | Blog | Sign up | Sign in

PUBLIC mininet / mininet | Star 468 | Fork 204

Home | Pages | History

Link modeling using ns 3

Page History | Clone URL

Contents

Relationship to ns-2

ns-3 is a new simulator, without backward compatibility

Similarities to ns-2:

- C++ software core
- GNU GPLv2 licensing
- ported ns-2 models: random variables, error models, OLSR, Calendar Queue scheduler

Differences:

- Python scripting (or C++ programs) replaces OTcl
- most of the core rewritten
- new animators, configuration tools, etc. are in work
- ns-2 is no longer actively maintained/supported

FAQs

- Does ns-3 have a Windows version?
  - Yes, for Visual Studio 2012
  - http://www.nsnam.org/wiki/Ns-3_on_Visual_Studio_2012
- Does ns-3 support Eclipse or other IDEs?
  - Instructions have been contributed by users
  - http://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3
- Is ns-3 provided in Linux or OS X package systems (e.g. Debian packages)?
  - Not yet
- Does ns-3 support NRL protolib applications?
  - Not yet

Summarizing

- ns-3 models are written in C++ and compiled into libraries
  - Python bindings are optionally created
- ns-3 programs are C++ executables or Python programs that call the ns-3 public API and can call other libraries
- ns-3 is oriented towards the command-line
- ns-3 uses no domain specific language
- ns-3 is not compatible with ns-2

Finding documentation and code

Resources

Web site:

http://www.nsnam.org

Mailing lists:

https://groups.google.com/forum/#!forum/ns-3-users

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

Wiki:

http://www.nsnam.org/wiki/

Tutorial:

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

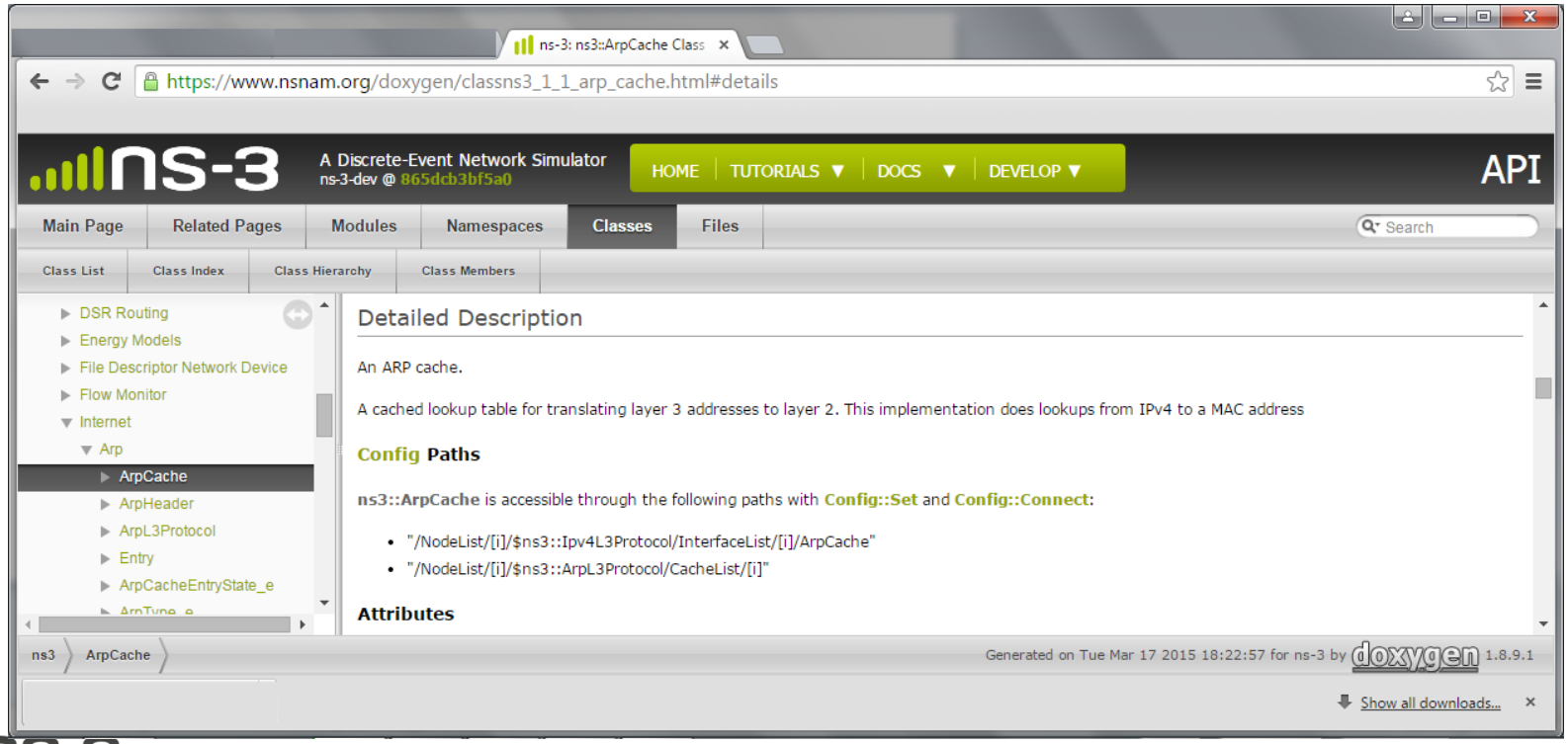
IRC: #ns-3 at freenode.net

Suggested steps

- Work through the ns-3 tutorial
- Browse the source code and other project documentation
  - manual, model library, Doxygen, wiki
  - ns-3 Consortium tutorials (May 2014)
    - https://www.nsnam.org/consortium/activities/training/
- Ask on ns-3-users mailing list if you still have questions
  - We try to answer most questions

APIs

- Most of the ns-3 API is documented with Doxygen
  - https://www.nsnam.org/doxygen



Reading existing code

- Much insight can be gained from reading ns-3 examples and tests, and running them yourselves
- Many core features of ns-3 are only demonstrated in the core test suite (src/core/test)
- Stepping through code with a debugger is informative
  - callbacks and templates make it more challenging than usual

Contributing

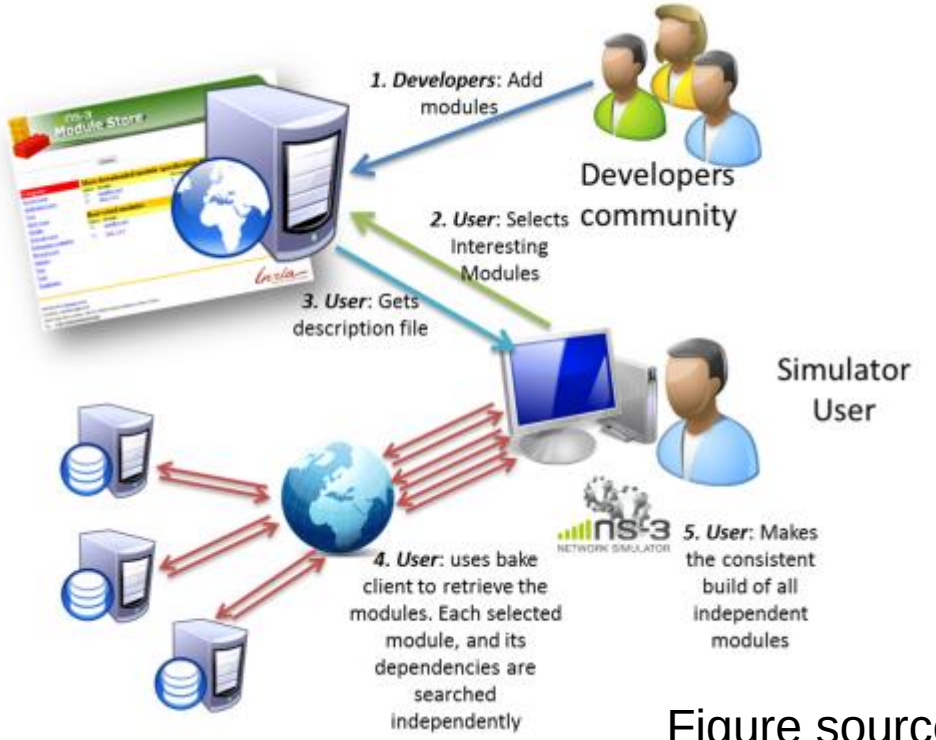
- Any amount of help is appreciated!
  - Reporting stale documentation to webmaster@nsnam.org
  - Contributing small patches
  - Writing new documentation
  - Reporting bugs
  - Fixing bugs
  - Reviewing code of others
  - Contributing new code
  - Becoming a maintainer

Development Priorities

- Software modularity and long-term maintenance
- Improved integration of direct code execution
- Improved integration with container-based and testbed-based experiment infrastructures
- Simulation-based experiment management
- Usability

Modularity

- Open source project maintains a (more stable) core
- Models migrate to a more federated development process



"bake" tool (Lacage and Camara)

Components:

- build client
- "module store" server
- module metadata

Figure source: Daniel Camara

Scope and capabilities

Current models

devices

- bridge
- mesh
- csma
- spectrum
- emu
- tap-bridge
- point-to-point
- uan
- lte
- virtual-net-device
- lr-wpan
- wifi
- wimax

- applications
- internet (IPv4/v6)
- sixlowpan
- wave
- network
- core

- energy
- mpi
- mobility
- propagation
- buildings

protocols

- aodv
- dsdv
- dsr
- olsr
- click
- nix-vector-routing
- openflow

utilities

- visualizer
- config-store
- flow-monitor
- netanim
- stats
- topology-read
- BRITE

WiFi model

- 802.11 a/b/g/p and some parts of n (SISO)
- Infrastructure and ad-hoc modes
- DCF implementation (Basic + RTS/CTS)
- QoS support (EDCA only)
- 802.11n (block ACK, MSDU and MPDU aggregation)
- Energy awareness
- Power and rate adaptation algorithms
- Vehicular (802.11p and WAVE)

LTE model

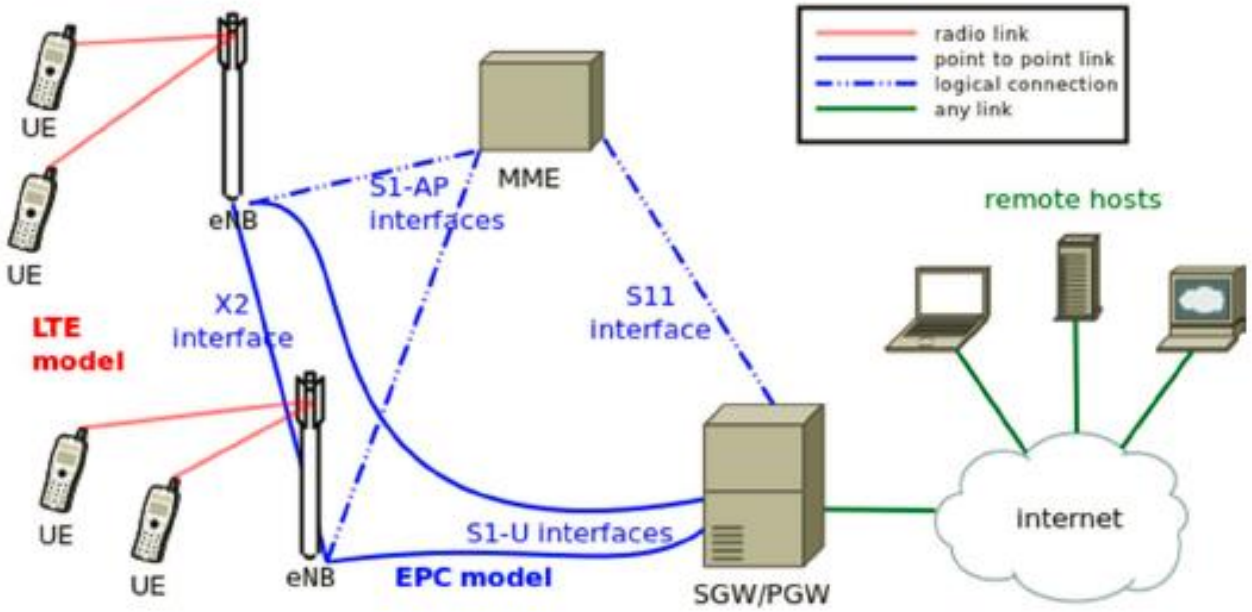
- A product-oriented simulator designed around an industrial API:
  - the Small Cell Forum MAC Scheduler Interface Specification
- Allows testing of real code in the simulation
- Accurate model of the LTE/EPC protocol stack
- Specific Channel and PHY layer models for LTE macro and small cells

LTE overview

- DL & UL LTE MAC Schedulers
- Radio Resource Management Algorithms
- Inter-cell interference coordination solutions
- Load Balancing and Mobility Management
- Heterogeneous Network (HetNets) solutions
- End-to-end QoE provisioning
- Multi-RAT network solutions
- Cognitive LTE systems

LTE overview

LENA model overview



Propagation models

- Several loss models available:
  - Friis
  - TwoRayGround
  - LogDistance
  - ThreeLogDistance
  - Jakes
  - OkumaraHata
  - FixedRss
  - etc.

Contributed code and associated projects

NS-3 based Named Data Networking (NDN) simulator
ndnSIM: NDN, CCN, CCNx, content centric networks

Overall ndnSIM documentation

ndnSIM ndnSIM API next

Next topic
Introduction

This Page
Show Source

Quick search
Go
Enter search terms or a module, class or function name.

Welcome to ndnSIM NS-3 based NDN simulator

We invite you to join our mailing list to see and participate in discussions about ndnSIM implementation and simulations in general (mailing list archives).

Contents:

- Introduction
  - More documentation
  - Support
  - A very short guide to the code
  - Logging
- Getting Started
  - Portability
  - Requirements
  - Downloading ndnSIM source
  - Compiling and running ndnSIM
- ndnSIM helpers
  - StackHelper
    - Routing
      - Manually routes
      - Global routing controller
      - Default routes
    - Content Store
    - Pending Interest Table
    - Forwarding strategy
  - AppHelper
- Content Store
  - Simple content stores
    - Least Recently Used (LRU) (default)
    - First-In-First-Out (FIFO)
    - Random
  - Content stores with entry lifetime tracking
    - Least Recently Used (LRU)
    - First-In-First-Out (FIFO)
    - Random
  - Content stores respecting freshness field of ContentObjects
    - Least Recently Used (LRU)
    - First-In-First-Out (FIFO)
    - Random

mptcp-ns3
implement multipath TCP on ns-3

Project Home | Downloads | Wiki | Issues | Source

Summary | Updates | People

Project Information

Starred by 4 users
Activity Medium
Project feeds

Code license
GNU Lesser GPL

Labels
Academic, ns-3, MPTCP, TCP, CPlusPlus, Study, MultipathTCP, multipath, Simulator

Members
chihan...@gmail.com

Links

External links
MPTCP IETF's Working Group
Multipath TCP in UCL
Multipath TCP in INL

Project description

The mptcp-ns3 project focuses on developing implementation of Multipath TCP on ns-3 for research purposes. The project implement the entire transport layer in ns-3.

Multipath TCP is an extension to TCP which aims to use multiple paths to handle a communication between two endpoints. MPTCP is the IETF working group to standardize Multipath TCP.

Please check the following URL for more information about multipath TCP. http://datatracker.ietf.org/wg/mptcp/charter/

Current Status

The current implementation is really close to the MPTCP specification:

- MPTCP options: *MPC* (Multipath Capable), *ADD* and *REMOVE* address, *JOIN*, etc.
- Congestion Control: *Fully Coupled*, *Uncoupled TCPs*, *Linked Increases*, *RTT Compensator*.
- Packet Reordering: None, Eifel, DSACK and F-RTO algorithms

Getting Started

Follow the instructions in the wiki page http://code.google.com/p/mptcp-ns3/wiki/MakeIt to successfully run simulations.

HOME | DEUTSCH | LEGALS | SITEMAP | KIT

KIT
Karlsruhe Institute of Technology

Decentralized Systems and Network Services Research Group - TM & SCC

Home
News
Teaching
Staff
Research
Publications
Traffic Telematics Junior Research Group
Miscellaneous / Software
Awareness Applet
JKad
JaaS Dispatcher
PhySimWiFi for NS-3
BitMON
JAAS Dispatcher
EMDV Visualizer
OvlVis
IEEE 802.11 Bugfix in ns-2.28
Overhaul of IEEE 802.11
How to contact us!

PhySimWiFi for NS-3

Contact: Jens Mittag, Stylianos Papanastasiou (CSS)
Project Group: DSN, Chalmers University of Technology - Signals and Systems (CSS)

Overview

PhySim-WiFi for NS-3 is a detailed and accurate implementation of the OFDM-based IEEE 802.11 standard within the popular network simulator NS-3. Compared to the default 802.11 PHY implementation of NS-3, which abstracts packets by considering only an average signal strength per paket und the length of the packet, the PhySim-WiFi implementation simulates the underlying signal processing steps of a transceiver down to the signal level, and introduces an increases accuracy for the decision whether a packet could be received correctly or not. At the same time, the new implementation allows to incoorporate more sophisticated channel models. For instance, due to the modeling of packets on a signal level, channel models can emulate multi-path effects much more accurately and are able to reflect Doppler effects and their impact on the physical layer signal processing algorithm.

The PhySim-WiFi implementation is a drop-in replacement of the default YansWifiPhy model, thus, it can be used with only minor modifications in the existing simulation code and the existing scenario setups.

For additional information and a changelog, please take a look at the

- PhySimWiFi Manual 1.1

Full package download

- NS-3.9-PhySimWiFi v1.1 (based on NS-3.9 and PhySim-WiFi) - August 19, 2011
- NS-3.9-PhySimWiFi v1.0 (based on NS-3.9 and PhySim-WiFi) - September 12, 2010

Patches for NS-3

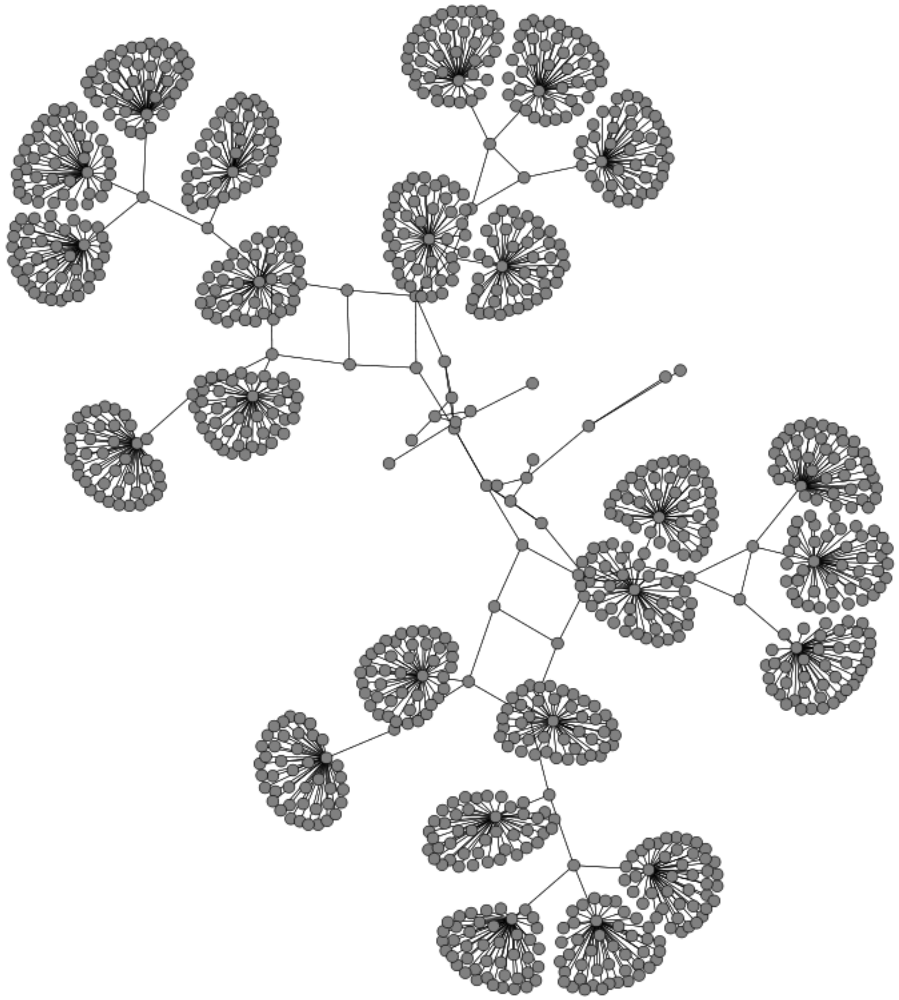
- PhySimWiFi v1.1 for NS-3.9 - August 19, 2011
- PhySimWiFi v1.1 for NS-3.9-PhySimWiFi v1.0 - August 19, 2011
- PhySimWiFi v1.0 for NS-3.9 - September 12, 2010

Visualization

PyViz overview

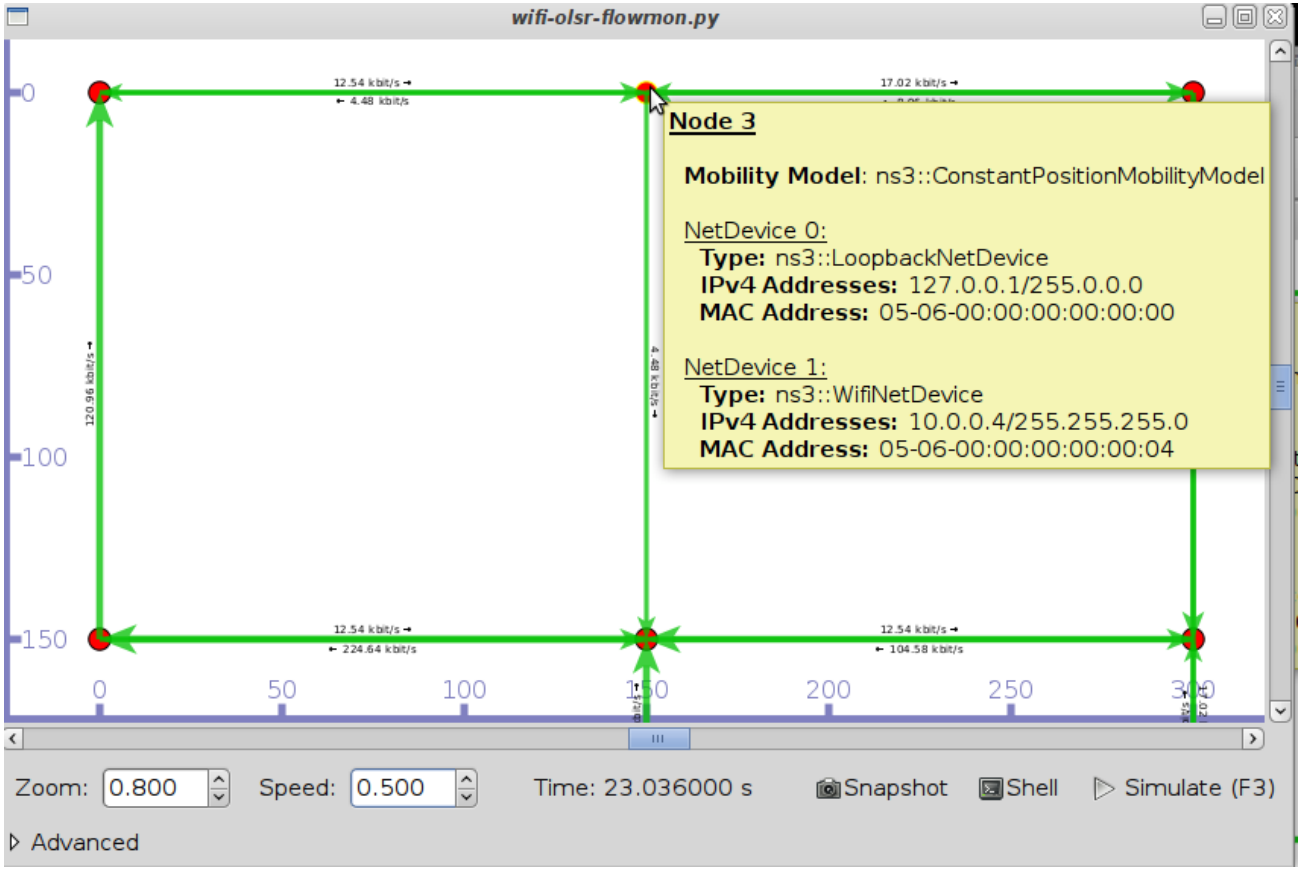
- Developed by Gustavo Carneiro
- Live simulation visualizer (no trace files)
- Useful for debugging
  - mobility model behavior
  - where are packets being dropped?
- Built-in interactive Python console to debug the state of running objects
- Works with Python and C++ programs

Pyviz screenshot (Graphviz layout)



Pyviz and FlowMonitor

- src/flow-monitor/examples/wifi-olsr-flowmon.py



Enabling PyViz in your simulations

- Make sure PyViz is enabled in the build

```
SQlite stats data output      : not enabled (library 'sqlite3' not found)
Tap Bridge                    : enabled
PyViz visualizer              : enabled
Use sudo to set suid bit      : not enabled (option --enable-sudo not selected)
```

- 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

G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.

FlowMonitor architecture

- Basic classes
  - FlowMonitor
  - FlowProbe
  - FlowClassifier
  - FlowMonitorHelper
- IPv6 coming in

ns-3.20 release

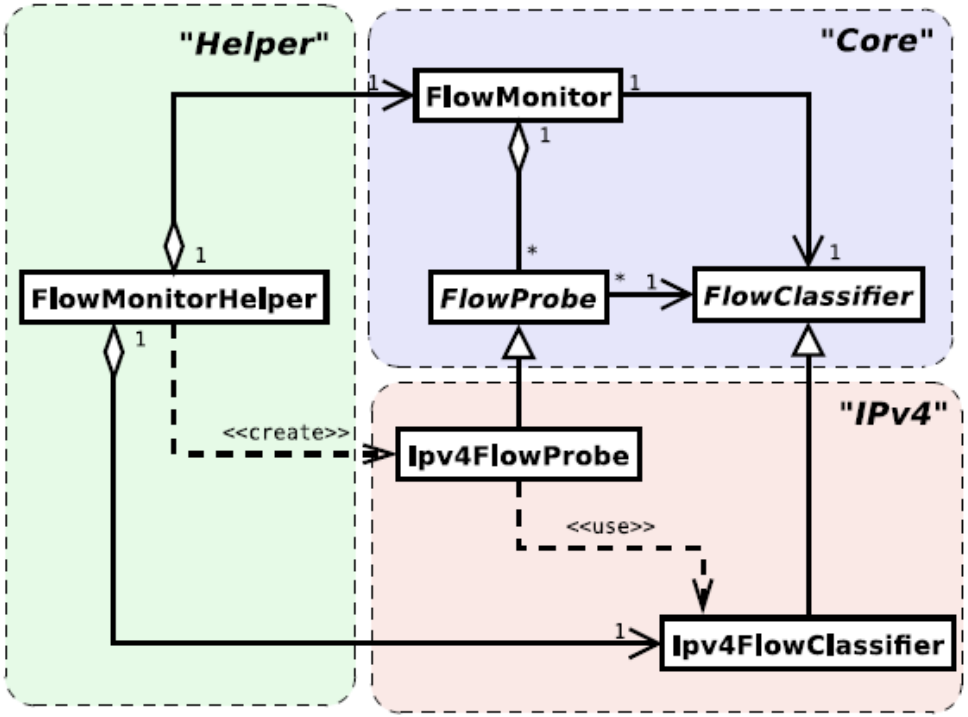
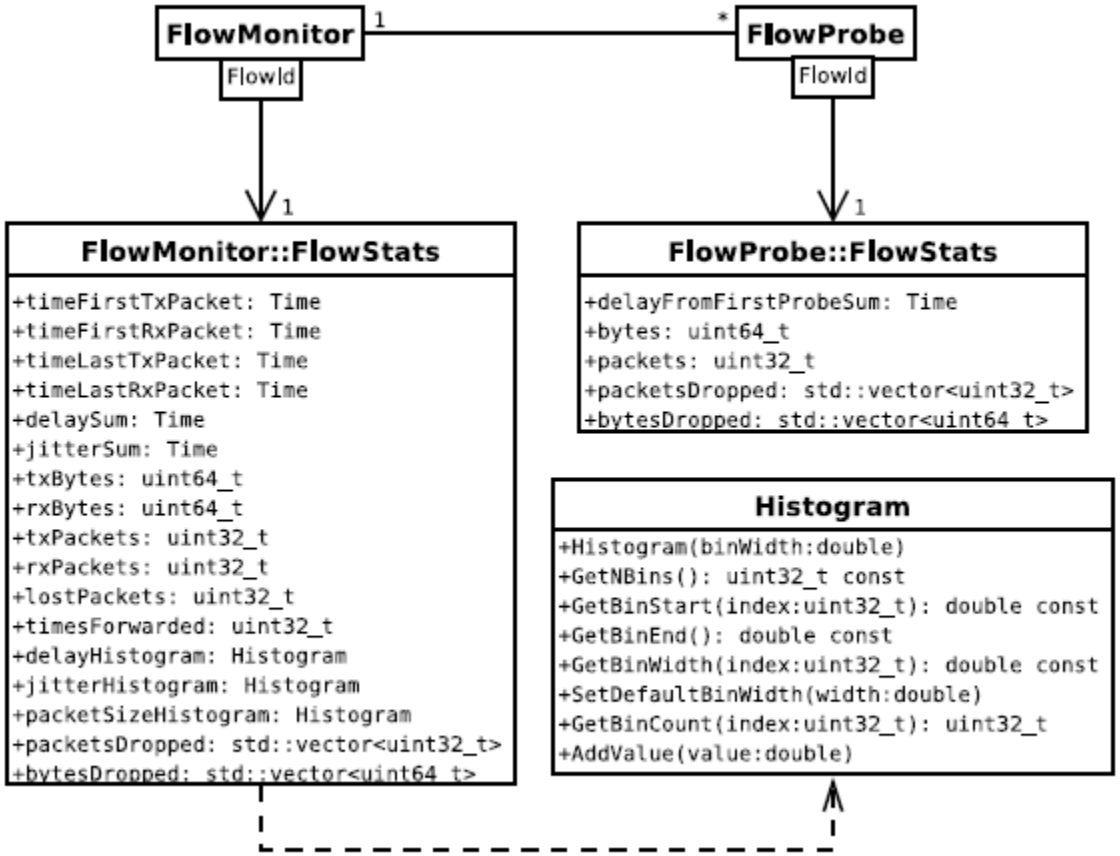


Figure credit: G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.

FlowMonitor statistics

- Statistics gathered



FlowMonitor configuration

- `example/wireless/wifi-hidden-terminal.cc`

```
// 8. Install FlowMonitor on all nodes
FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll ();

// 9. Run simulation for 10 seconds
Simulator::Stop (Seconds (10));
Simulator::Run ();

// 10. Print per flow statistics
monitor->CheckForLostPackets ();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i != stats.end (); ++i)
  {
    // first 2 FlowIds are for ECHO apps, we don't want to display them
    if (i->first > 2)
      {
        Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
        std::cout << "Flow " << i->first - 2 << " (" << t.sourceAddress << " -> " << t.destinationAddress << ")\n";
        std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
        std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
        std::cout << "  Throughput: " << i->second.rxBytes * 8.0 / 10.0 / 1024 / 1024  << " Mbps\n";
      }
  }
```

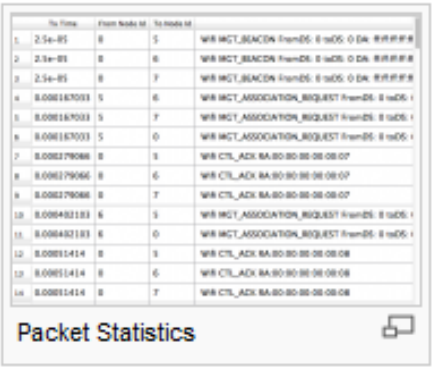
FlowMonitor output

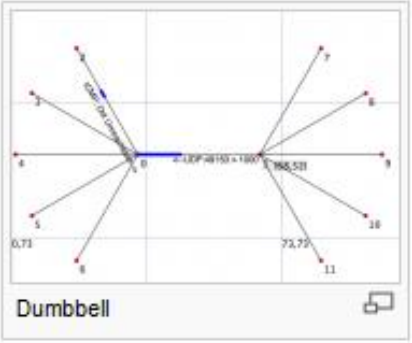
- This program exports statistics to stdout
- Other examples integrate with PyViz

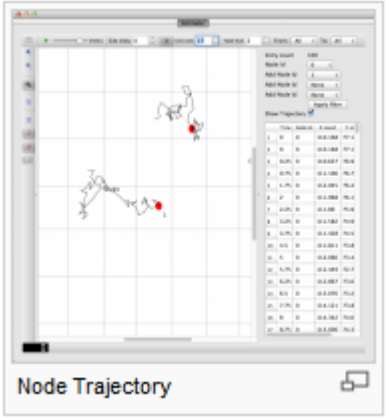
```
Hidden station experiment with RTS/CTS disabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes:   3847500
  Rx Bytes:   316464
  Throughput: 0.241443 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:   3848412
  Rx Bytes:   336756
  Throughput: 0.256924 Mbps
-----------------------------------------------
Hidden station experiment with RTS/CTS enabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes:   3847500
  Rx Bytes:   306660
  Throughput: 0.233963 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:   3848412
  Rx Bytes:   274740
  Throughput: 0.20961 Mbps
```

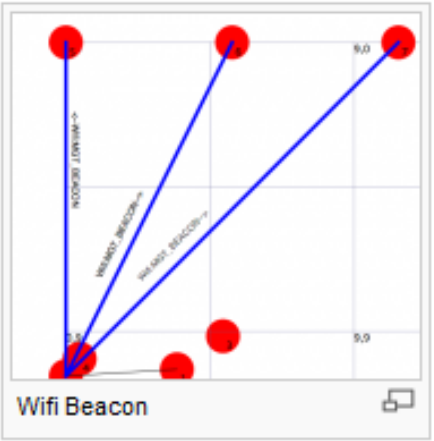
NetAnim

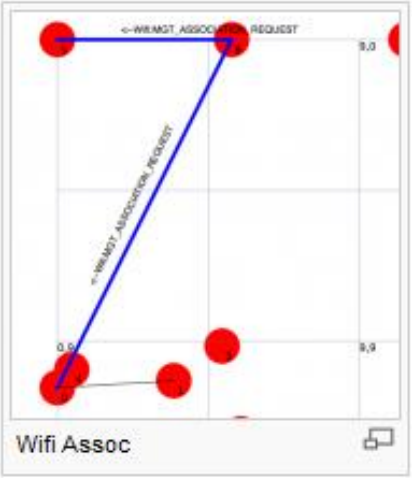
- "NetAnim" by George Riley and John Abraham

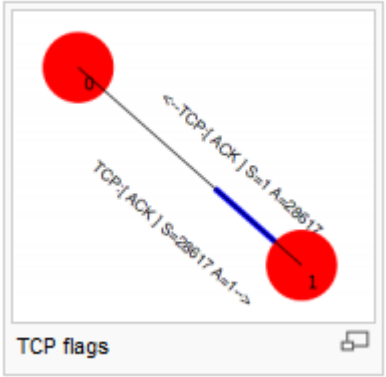

Packet Statistics


Dumbbell


Node Trajectory


Wifi Beacon


Wifi Assoc


TCP flags

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

Placeholder for netanim videos