ns-3 Training

Session 1: Monday 8:30am

ns-3 Annual Meeting
May 2014

Logistics

- Introductions
- Wireless access
- Wiki page:
  - http://www.nsnam.org/wiki/Training2014

ns-3 training goals

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

ns-3 training agenda

- Monday: Overview of ns-3
  - Session 1: Overview
  - Session 2: Core, basic models, I/O
  - Session 3: Tracing
  - Session 4: Writing new code
- Tuesday: Special topics
  - Session 1: WiFi and mobile network simulations
  - Session 2: LTE
  - Session 3: Parallel, distributed simulations
  - Session 4: DCE and Emulation

ns-3 training agenda: Monday morning

- 8:30-10h00: ns-3 overview
  - Overview of software and models
  - Running and understanding an existing example
  - Basic animation and visualization (netanim, flow monitor)
  - Integrating other tools and libraries

- 10:00-10:15: 15-minute coffee break
- 10:15-12:00: ns-3 core
  - object model
  - scheduler
  - callbacks
  - etc.

ns-3 training agenda: Monday afternoon

- 1:30-3:00: Tracing and data collection
  - Tracing subsystem in depth
  - Data collection framework

- 3:00-3:15: 15-minute coffee break
- 3:15-5:00: Writing new software, Q&A
  - Writing new examples and models

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) five of the past six summers

Acknowledgment of support

National Science Foundation
WHERE DISCOVERIES BEGIN

informatics mathematics

PLANETE

Georgia Tech

Google SUMMER OF CODE 2012

INESCPORTO
LABORATÓRIO ASSOCIADO

Information Sciences Institute

Goals of the consortium

- 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.

What is ns-3?

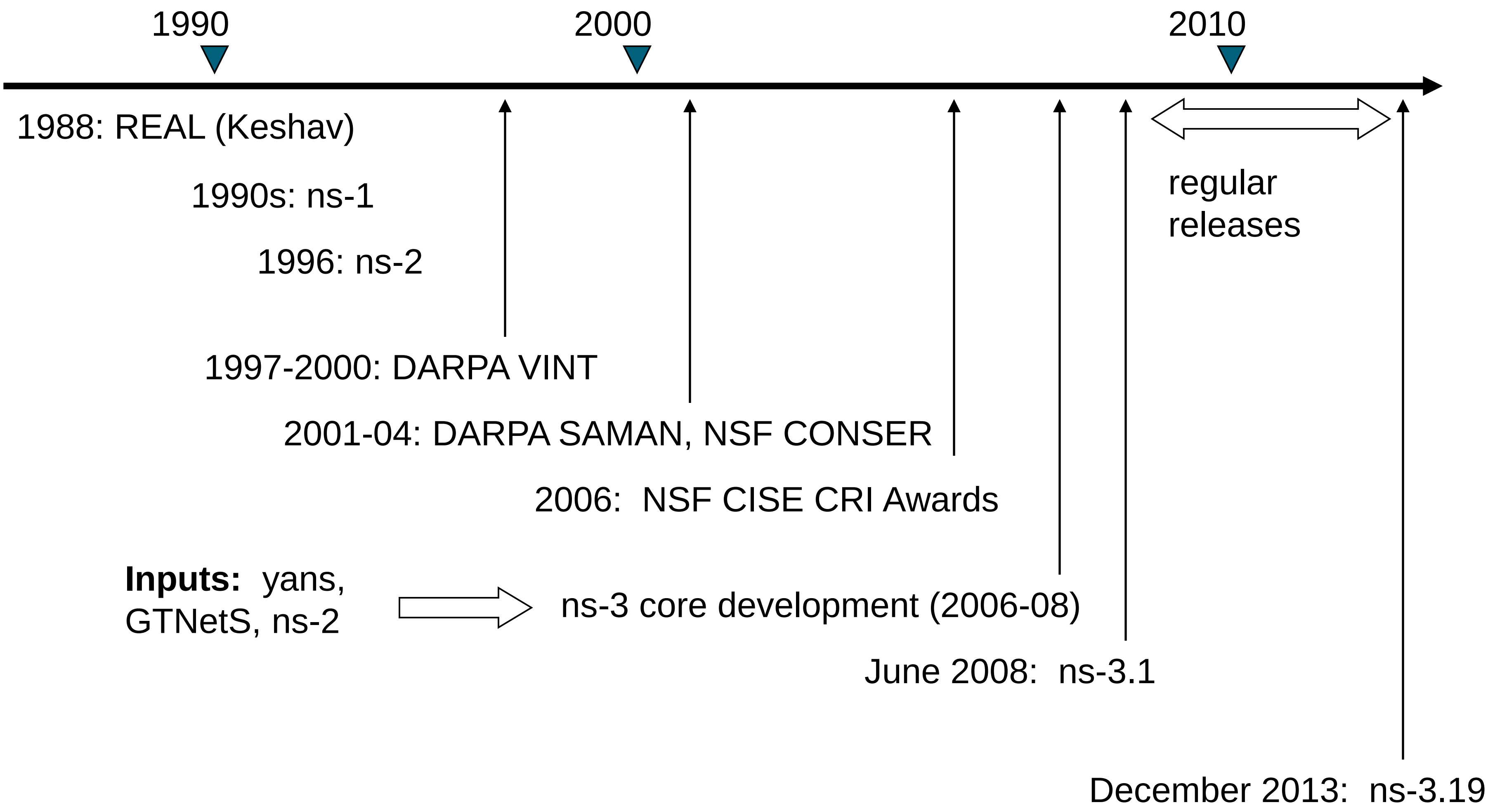
Discrete-event simulation basics

- 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() gets it all started
- Simulation stops at specific time or when events end

Preliminaries

- ns-3 is written in C++, with bindings available for Python
  - simulation programs are C++ executables or Python programs
  - ~300,000 lines of mostly 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

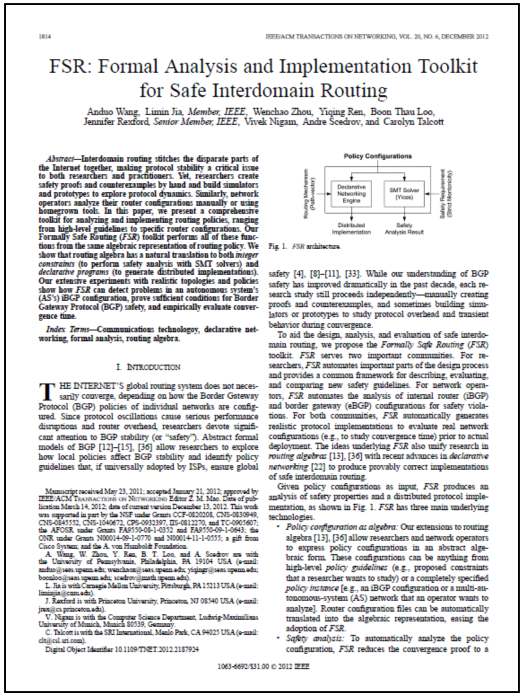
ns timeline



1990
2000
2010
1988: REAL (Keshav)
1990s: ns-1
1996: ns-2
1997-2000: DARPA VINT
2001-04: DARPA SAMAN, NSF CONSER
2006: NSF CISE CRI Awards
Inputs: yans, GTNetS, ns-2
ns-3 core development (2006-08)
June 2008: ns-3.1
regular releases
December 2013: ns-3.19

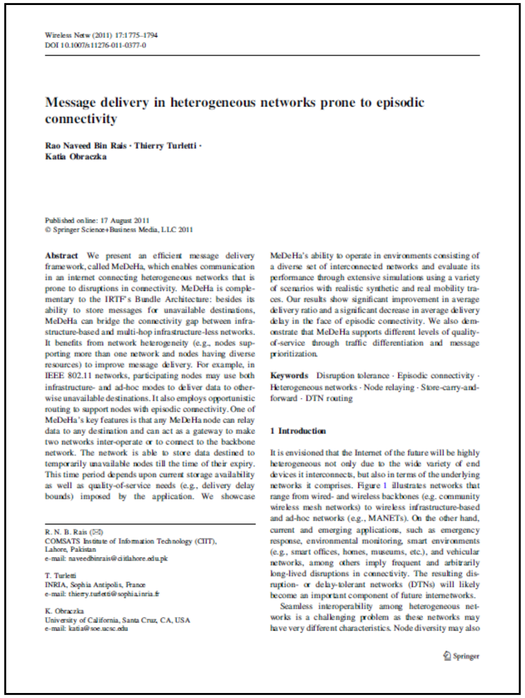
What have people done with ns-3?

- ~300 publications (March 2013)
  - search of 'ns-3 simulator' on IEEE and ACM digital libraries


1814 IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 20, NO. 6, DECEMBER 2012

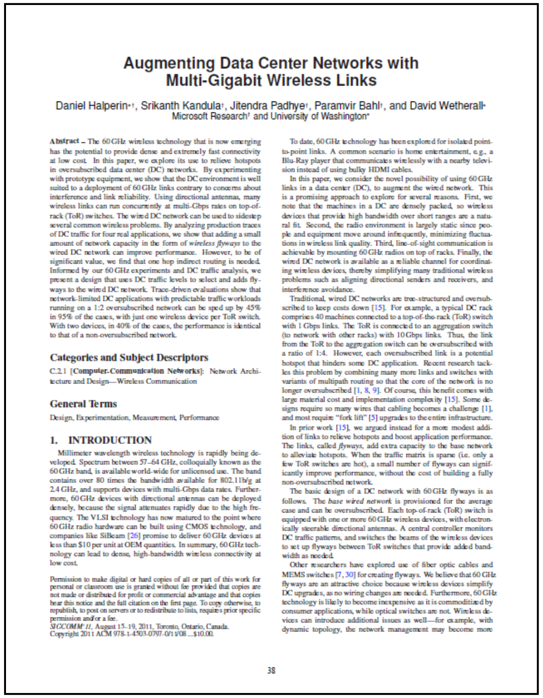
FSR: Formal Analysis and Implementation Toolkit for Safe Interdomain Routing

Anduo Wang, Limin Jia, Member, IEEE, Wenchao Zhou, Yiqing Ren, Boon Thau Loo, Jennifer Rexford, Senior Member, IEEE, Vivek Nigam, Andre Scedrov, and Carolyn Talcott


Wireless Netw (2011) 17:1775–1794
DOI 10.1007/s11276-011-0377-0

Message delivery in heterogeneous networks prone to episodic connectivity

Rao Naveed Bin Rais · Thierry Turletti · Katia Obraczka


Augmenting Data Center Networks with Multi-Gigabit Wireless Links

Daniel Halperin, Srikanth Kandula, Jitendra Padhye, Paramvir Bahl, and David Wetherall
Microsoft Research and University of Washington

What have people done with ns-3?

- Educational use (from ns-3 wiki)

Using ns-3 in Education

This page is a resource for learning about ns-3 as an educational tool for networking education.

Papers

The 2011 Sigcomm Education workshop had a paper regarding ns-3 use in the classroom:

- An Open-source and Declarative Approach Towards Teaching Large-scale Networked Systems Programming

Courses using ns-3

The following courses have used ns-3 as courseware or to support projects

- Georgia Tech. ECE 6110 Dr. George Riley, Spring 2013 (also Fall 2011, Fall 2010)
- The University of Kansas EECS 780, EECS 882, and EECS 983 Dr. James Sterbenz, 2010 – 2012
- UPenn CIS 553/TCOM 512 Dr. Boon Thau Loo, Fall 2010
- Aalto University Jose Costa-Requena and Markus Peuhkuri, Fall 2011
- Indian Institute of Technology Bombay Bhaskaran Raman, Autumn 2008
- University of Rijeka
  - RM2-InfUniRi, Dr. Mario Radovan and Vedran Miletić, Spring 2013, also Spring 2012
  - RM-RiTeh, Dr. Mladen Tomić and Vedran Miletić, Spring 2013

Other resources

- Lalith Suresh's Lab Assignments using ns-3 page.

Software introduction

- Download the latest release
  - `wget http://www.nsnam.org/releases/ns-allinone-3.19.tar.bz2`
  - `tar xjf ns-allinone-3.19.tar.bz2`
- Clone the latest development code
  - `hg clone http://code.nsnam.org/ns-3-allinone`

Q. What is "**hg clone**"?
A. Mercurial (http://www.selenic.com) is our source code control tool.

Software for ns-3 training

Two versions

1) ns-allinone package

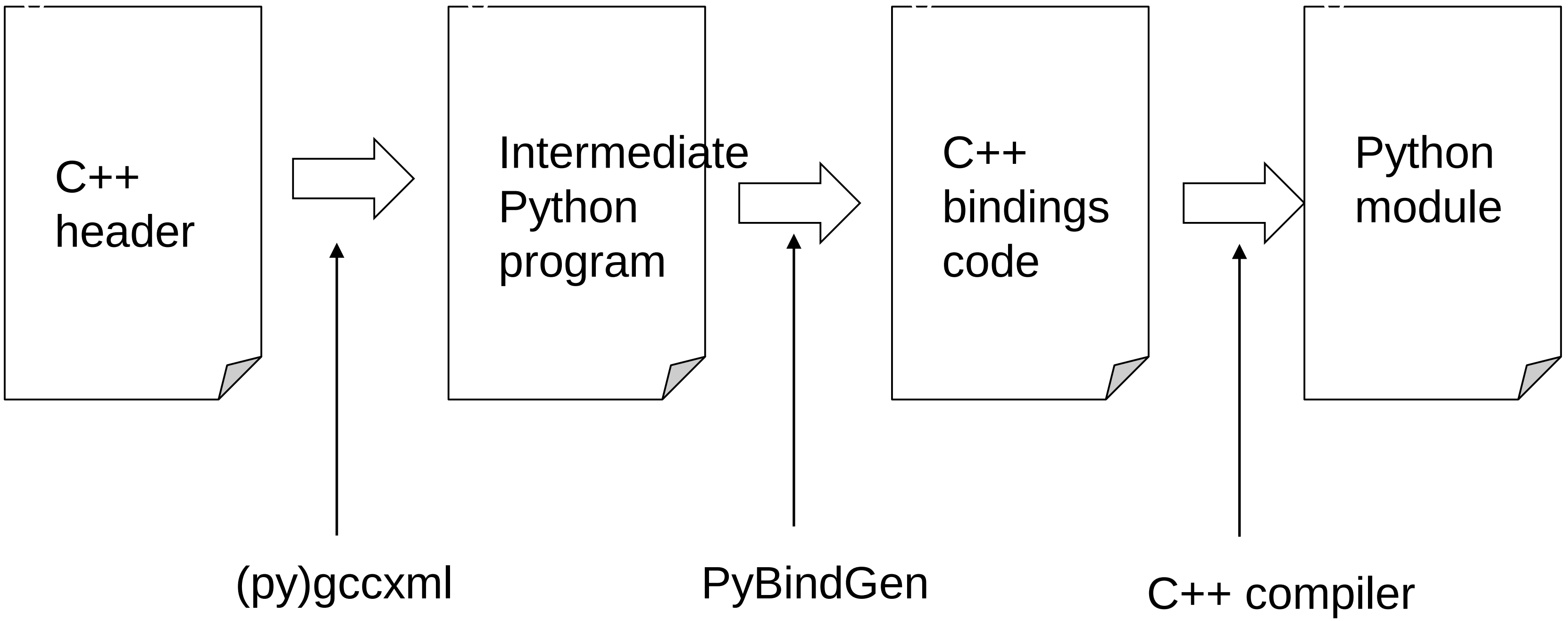
- `wget` https://www.nsnam.org/tutorials/consortium14/ns-allinone-3.20.training.tar.bz2
- `tar xjf ns-allinone-3.20.training.tar.bz2`

2) bake package

- `wget` https://www.nsnam.org/tutorials/consortium14/bake-ns-3.20.training.tar.bz2
- `tar xjf bake-ns-3.20.training.tar.bz2`

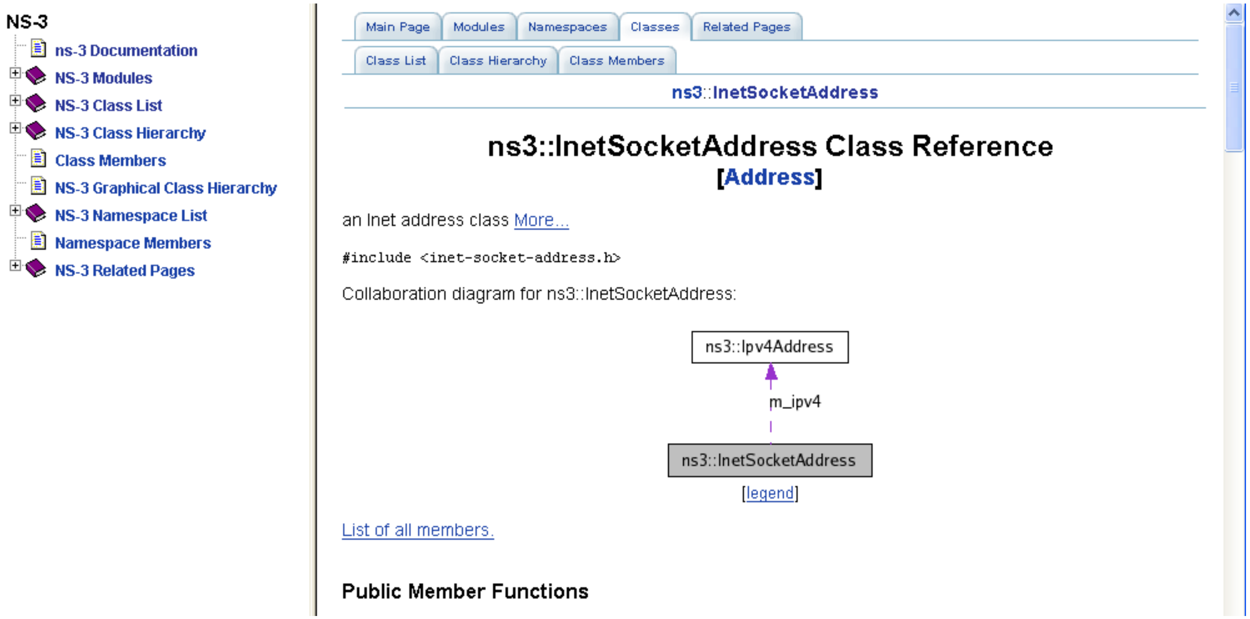
Python bindings

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



APIs

- Most of the ns-3 API is documented with Doxygen
  - http://www.stack.nl/~dimitri/doxygen/



mixed-wireless example

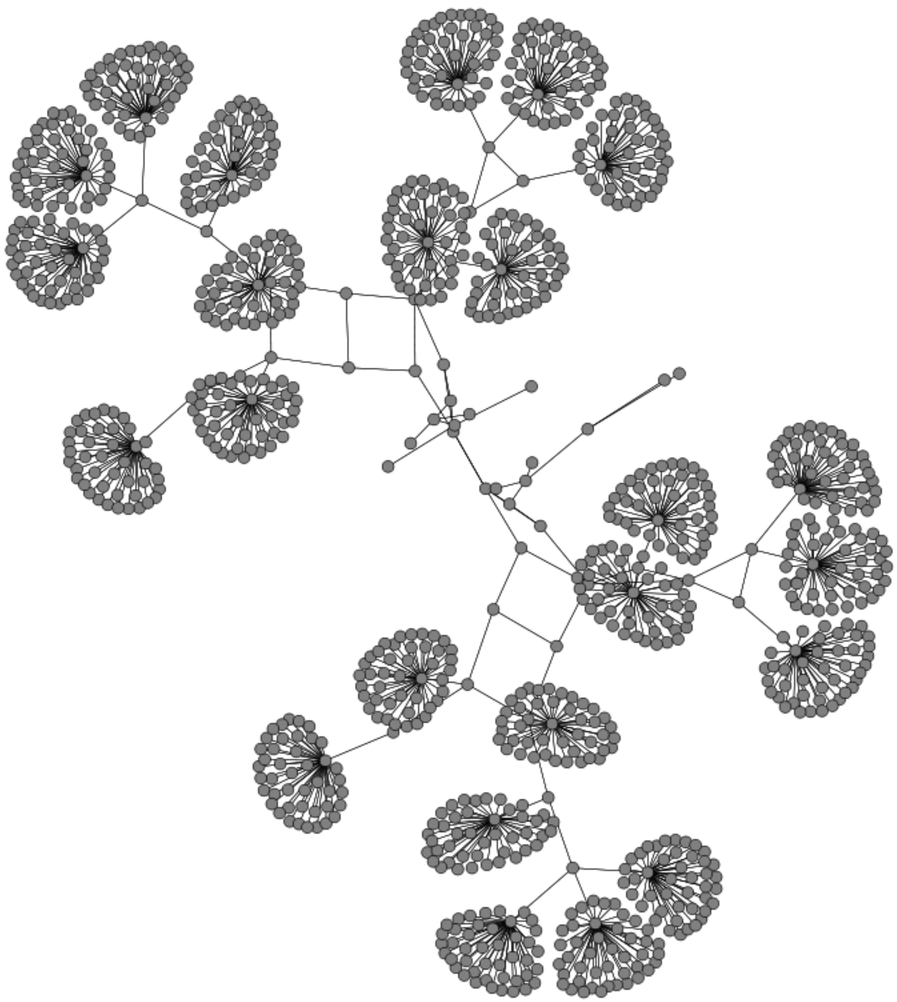
- Placeholder slide

- `./waf --run mixed-wireless`
- `./waf --pyrun examples/wireless/mixed-wireless.py`

PyViz overview

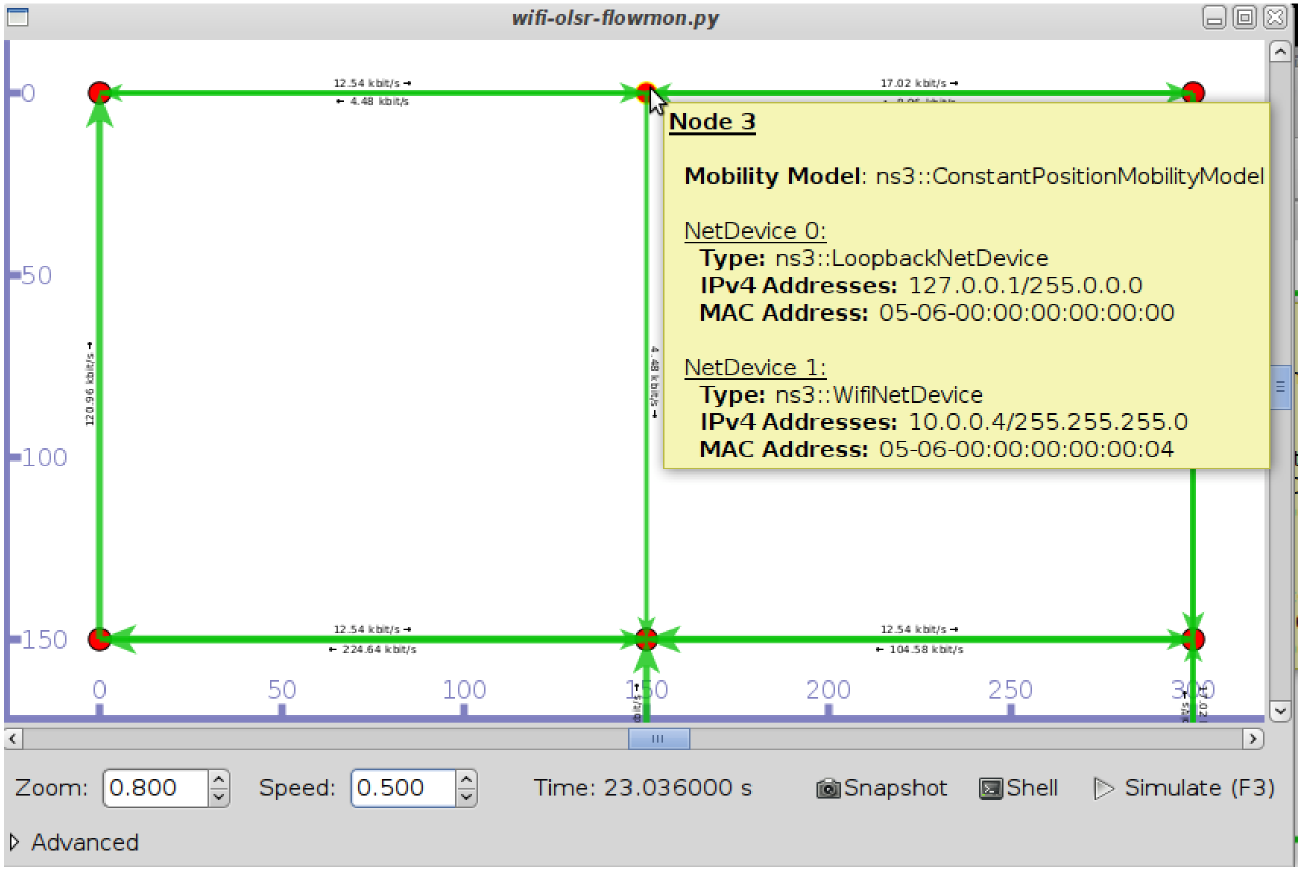
- Developed by Gustavo Carneiro
- Live simulation visualizer (no trace files)
- Useful for debugging
  - mobility model behavior
  - where are packets being dropped?
- Built-in interactive Python console to debug the state of running objects
- Works with Python and C++ programs

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

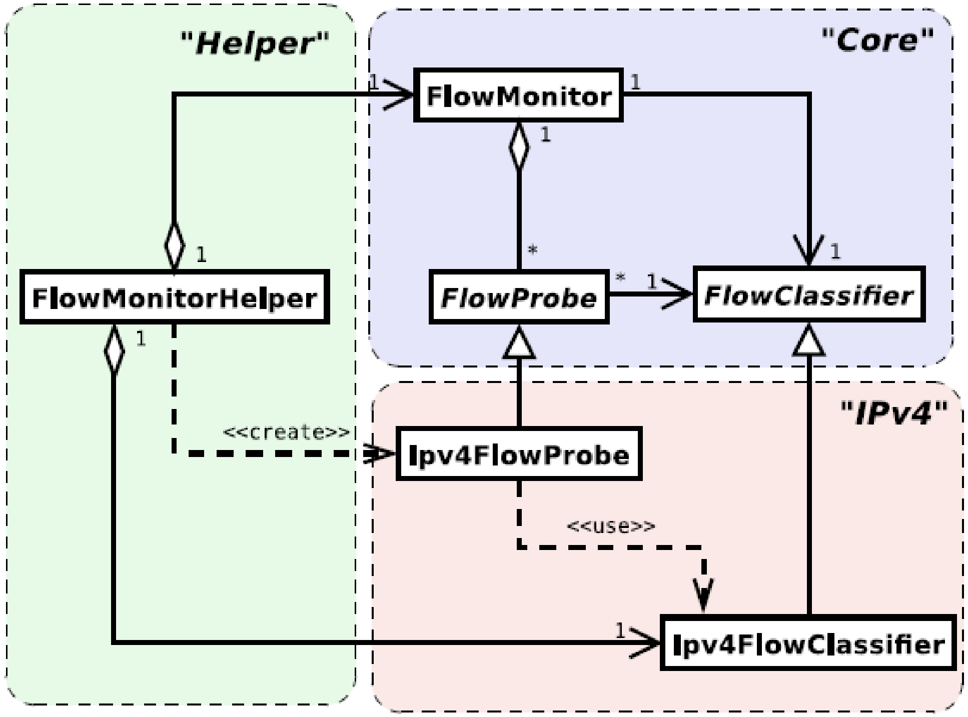
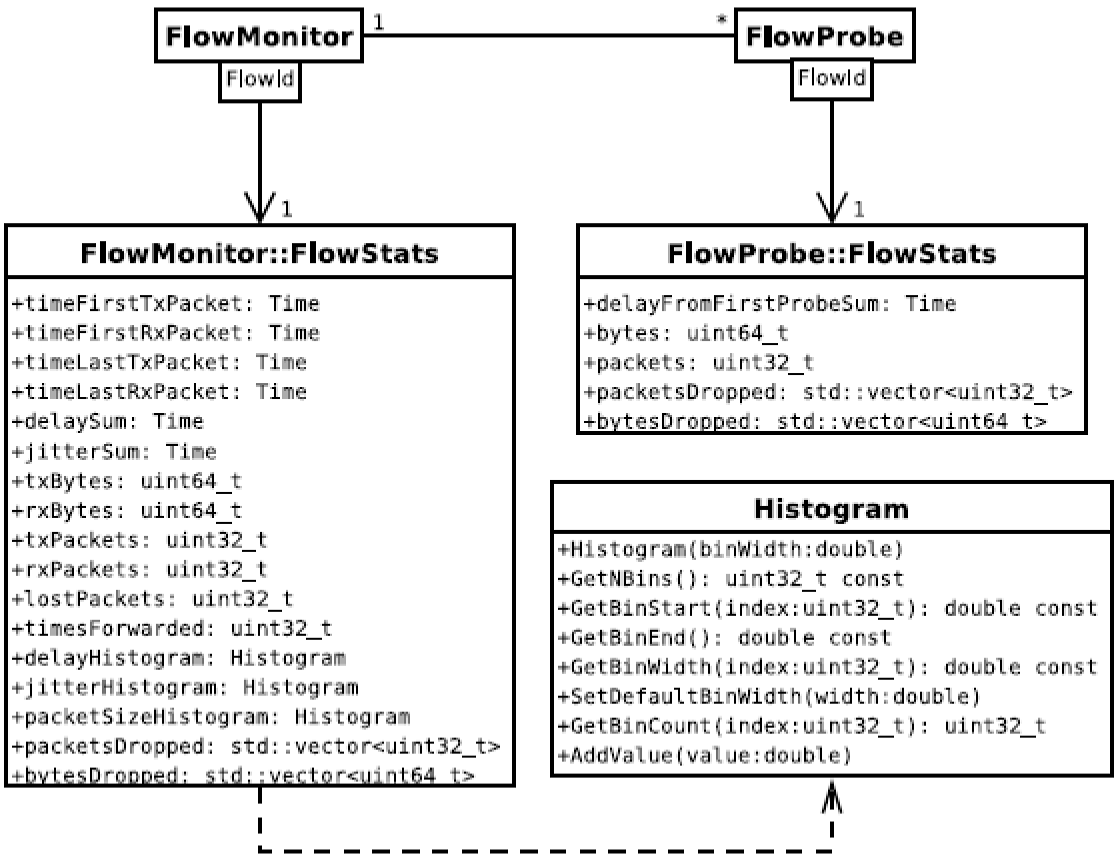


Figure credit: G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.

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";
      }
  }
```

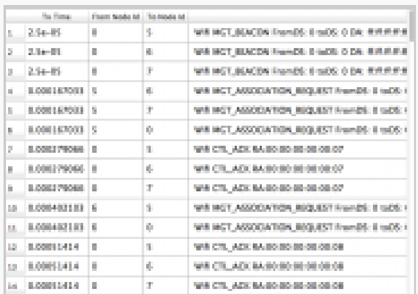
FlowMonitor output

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

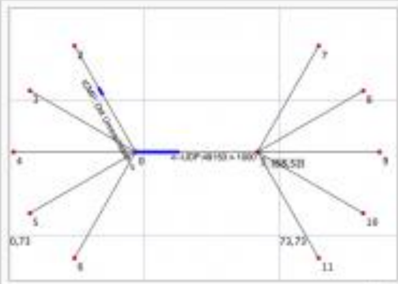
```
Hidden station experiment with RTS/CTS disabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes:   3847500
  Rx Bytes:   316464
  Throughput: 0.241443 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:   3848412
  Rx Bytes:   336756
  Throughput: 0.256924 Mbps
------------------------------------------------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

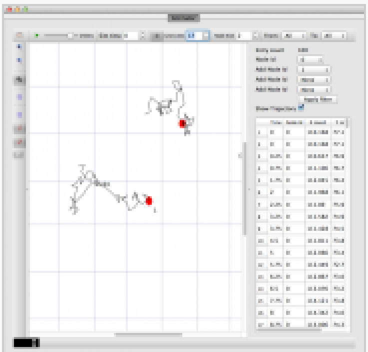
- "NetAnim" by George Riley and John Abraham



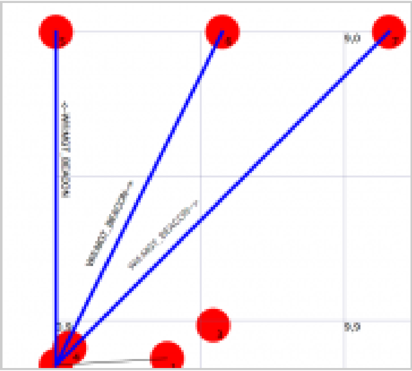
Packet Statistics



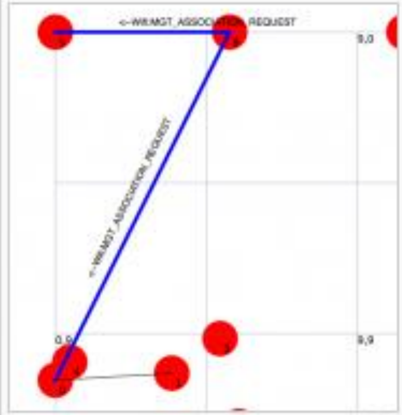
Dumbbell



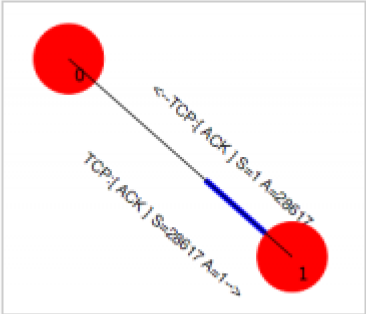
Node Trajectory



Wifi Beacon



Wifi Assoc



TCP flags

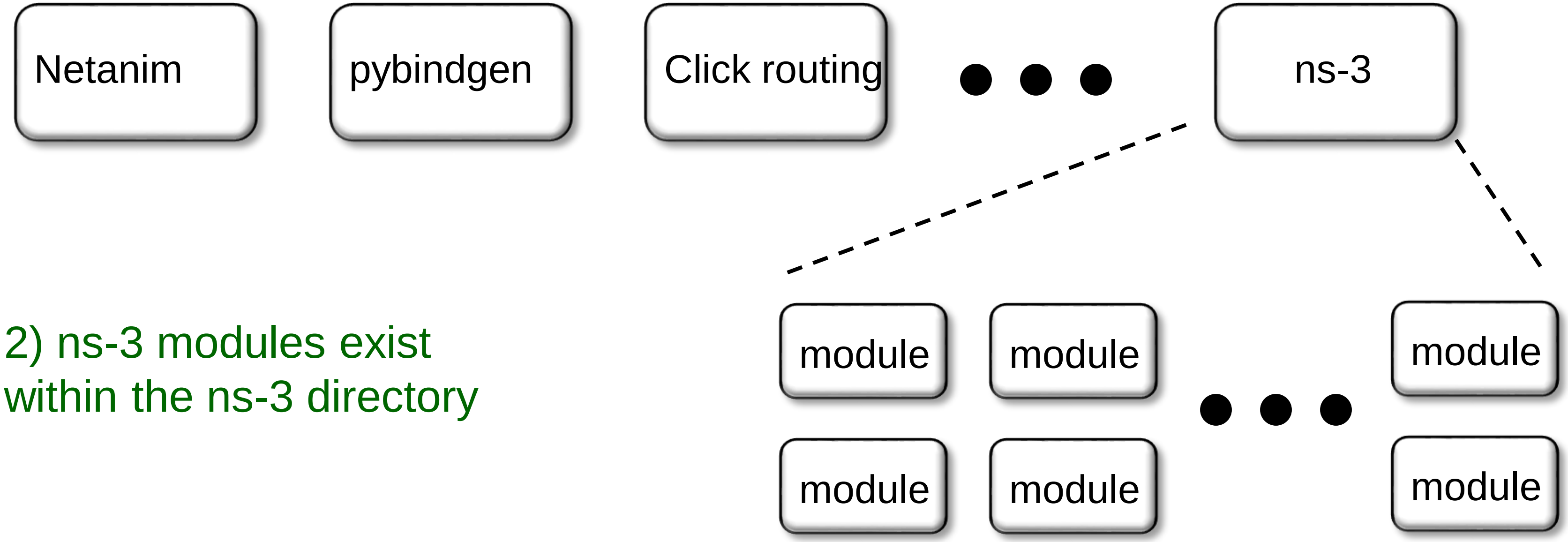
NetAnim key features

- 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

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

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

- energy
- mpi
- mobility
- propagation

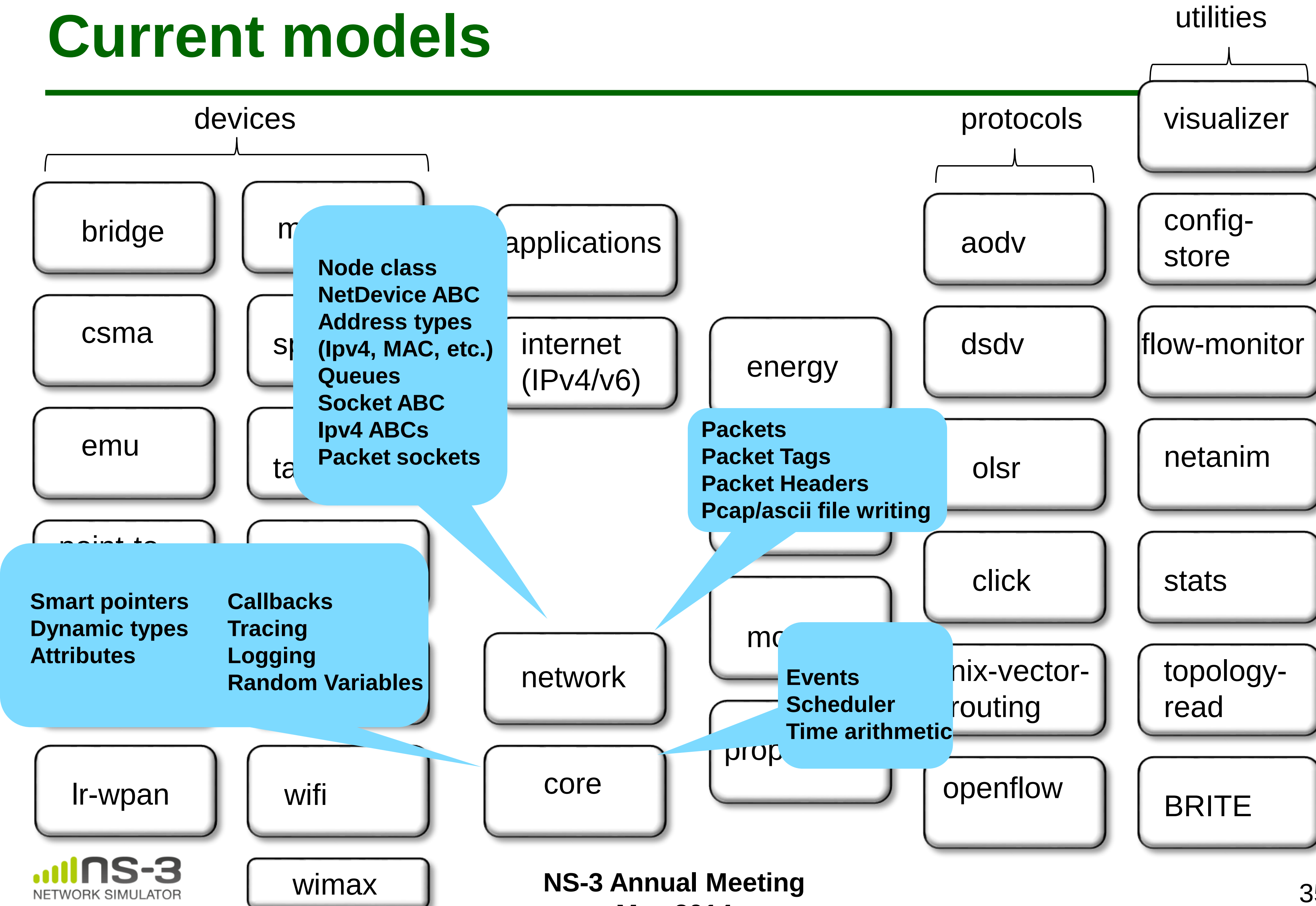
protocols

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

utilities

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

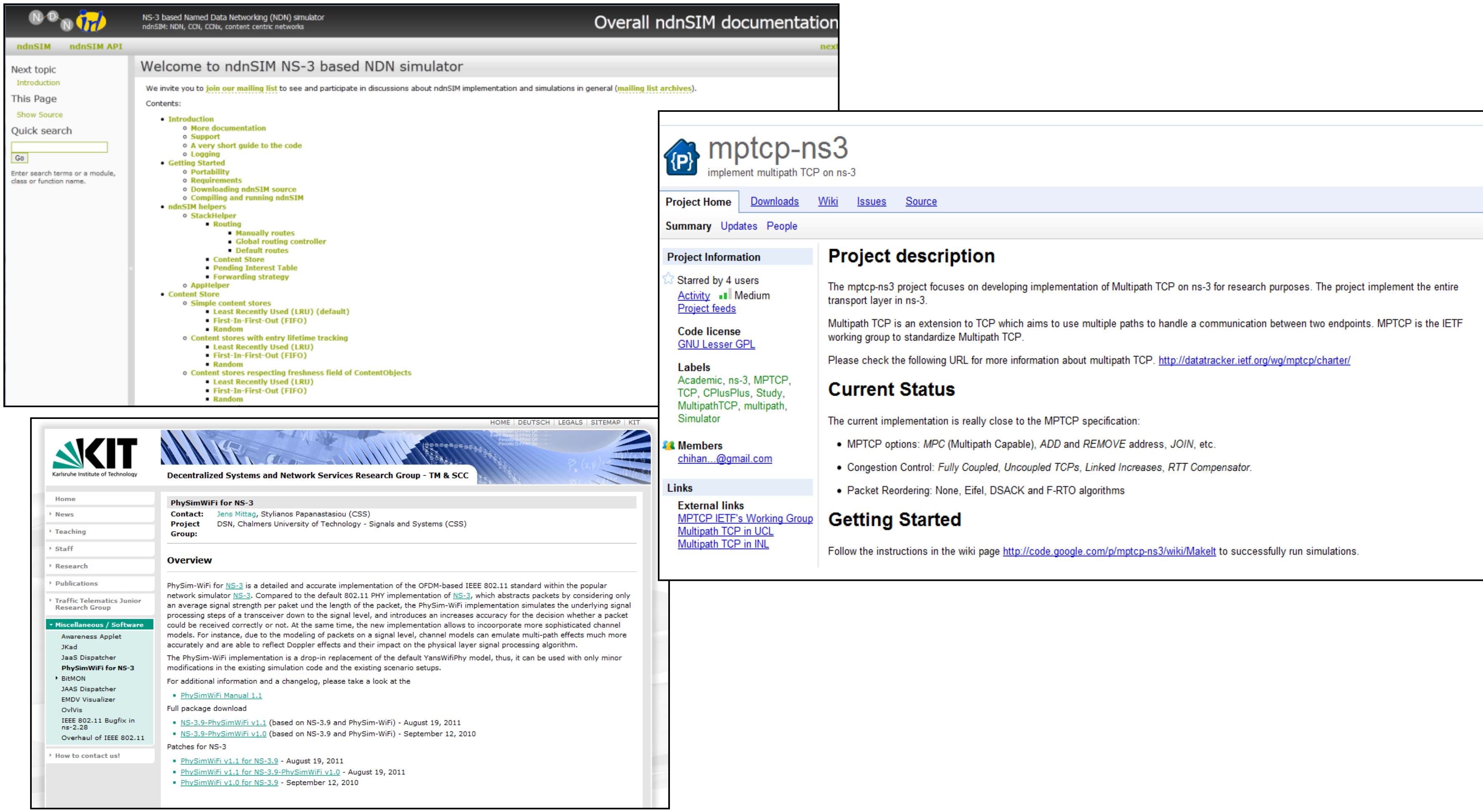
Current models



Module organization

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

Contributed code and associated projects



Integrating other tools and libraries

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

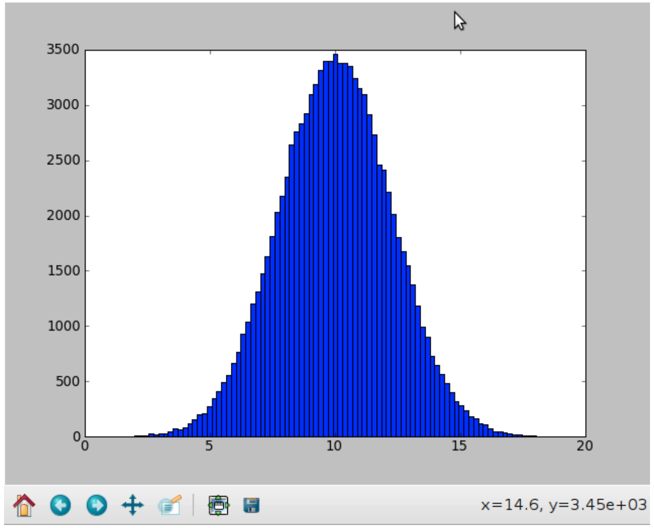
```
    uint32_t pktsRecvd = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
    dataset.Add (rss, pktsRecvd);
  }

gnuplot.AddDataset (dataset);
```

Add data to dataset

Add dataset to plot

Matplotlib

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

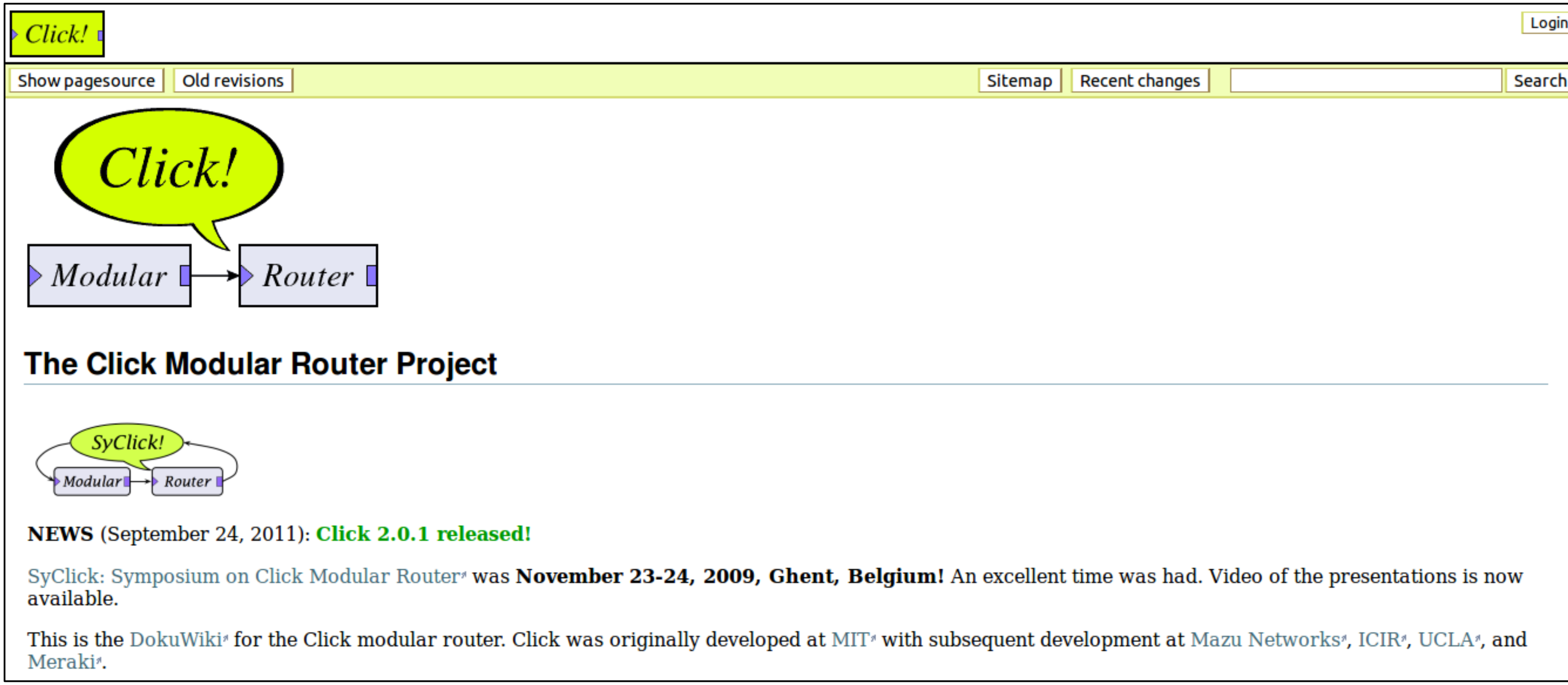



```
# Demonstrate use of ns-3 as a random number generator integrated
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial

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()
```

Other libraries

- ns-3 supports additional libraries (click, openflow, nsc)
- ns-3 has optional libraries (libxml2, gsl, mysql)
- both are typically enabled/disabled through the wscript
- users are free to write their own Makefiles or wscripts to do something special

Click Modular Router



Click!

Login

Show pagesource | Old revisions | Sitemap | Recent changes | Search

Click!

Modular → Router

The Click Modular Router Project

SyClick!

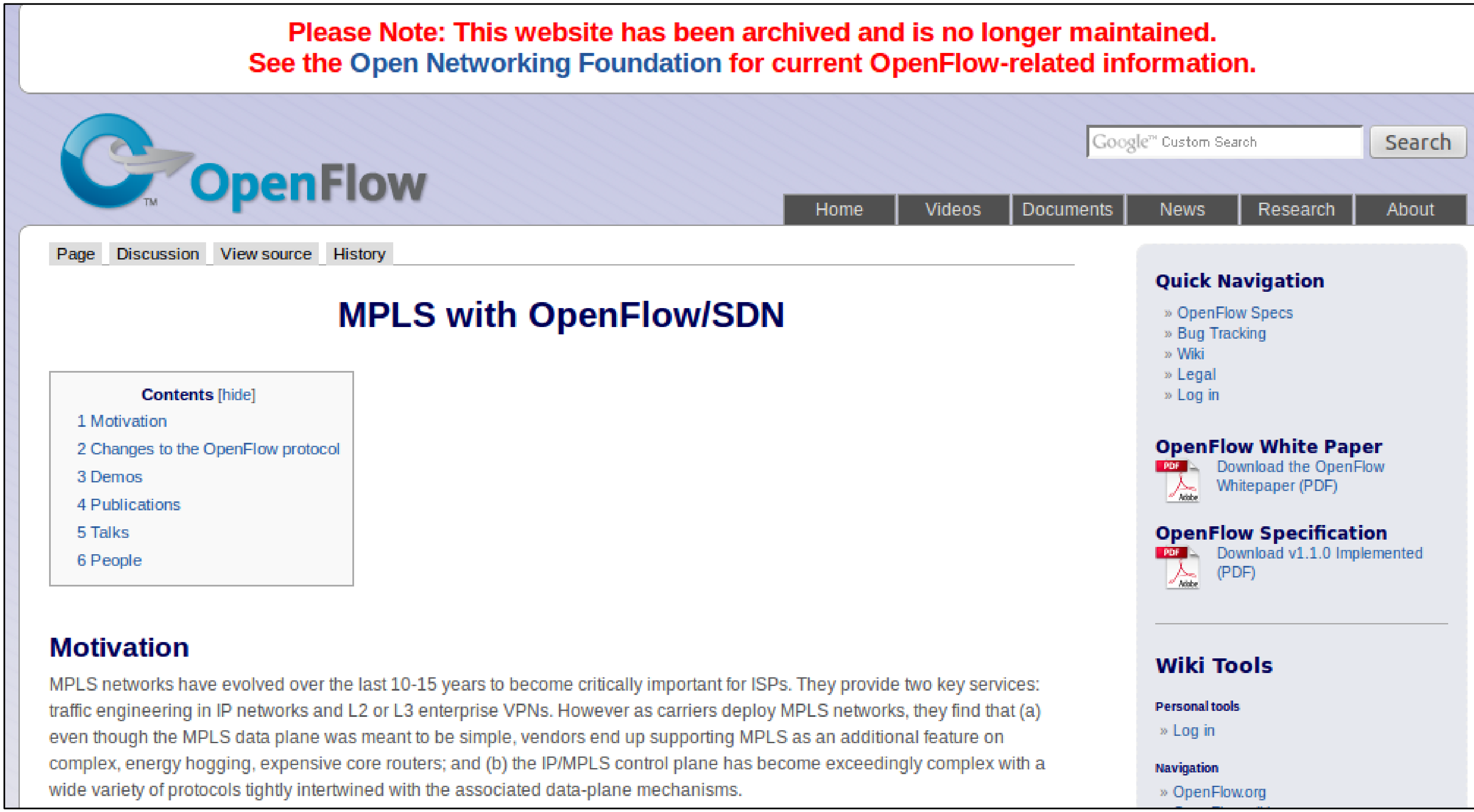
Modular → Router

NEWS (September 24, 2011): **Click 2.0.1 released!**

SyClick: Symposium on Click Modular Router was **November 23-24, 2009, Ghent, Belgium!** An excellent time was had. Video of the presentations is now available.

This is the DokuWiki for the Click modular router. Click was originally developed at MIT with subsequent development at Mazu Networks, ICIR, UCLA, and Meraki.

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

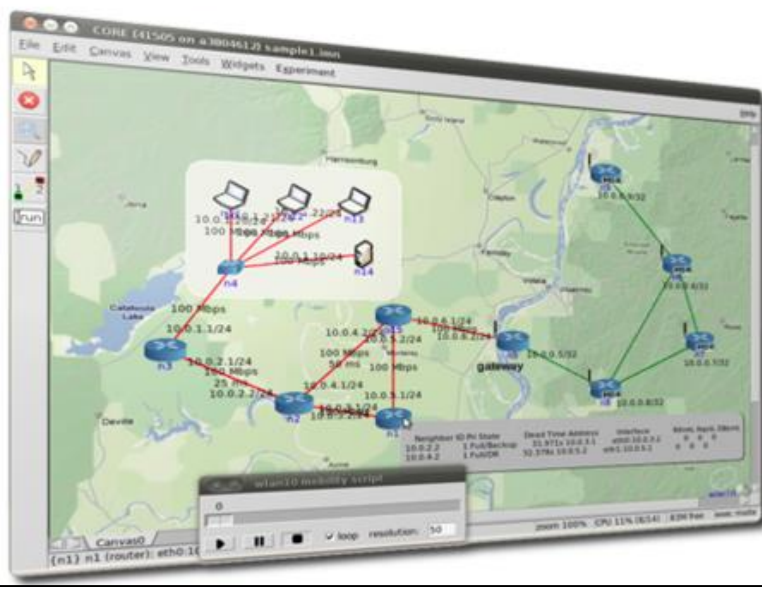
Focus Areas | Projects | Products | Organization

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

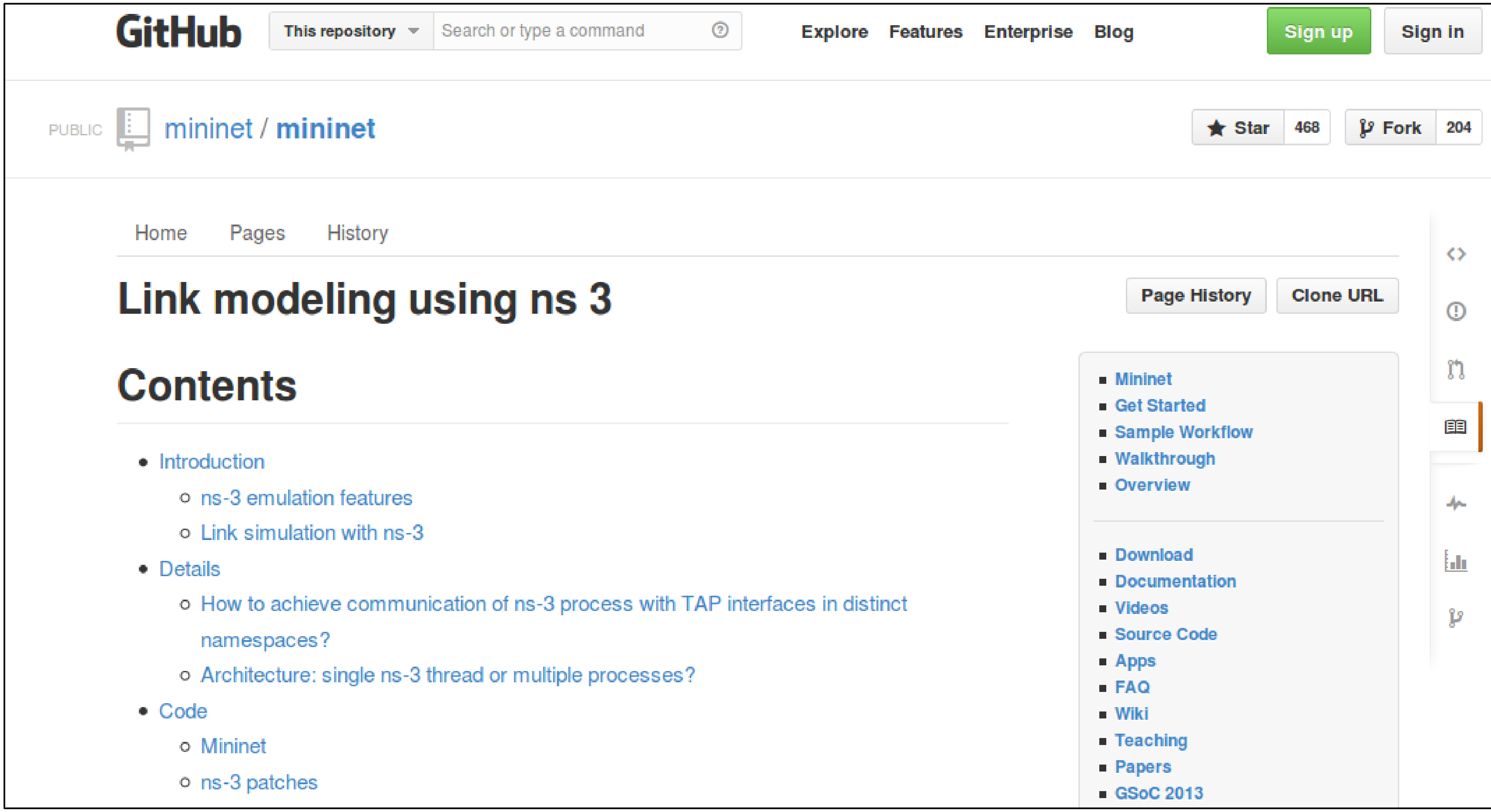
Common Open Research Emulator (CORE)

- 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.

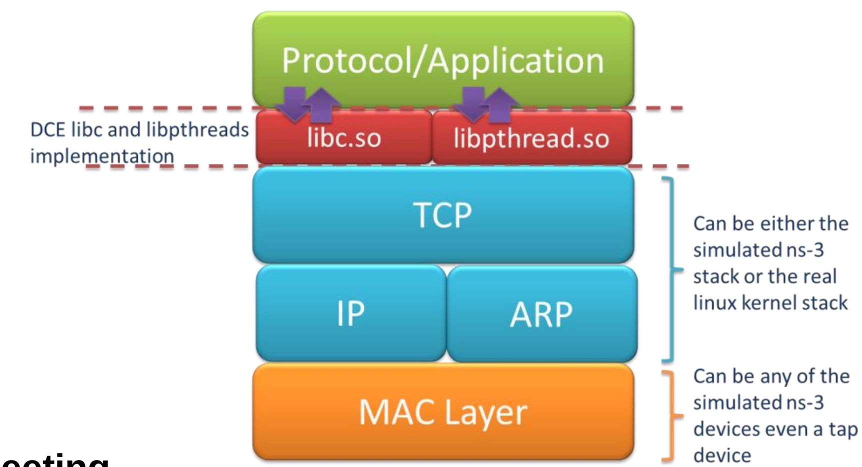


mininet emulator

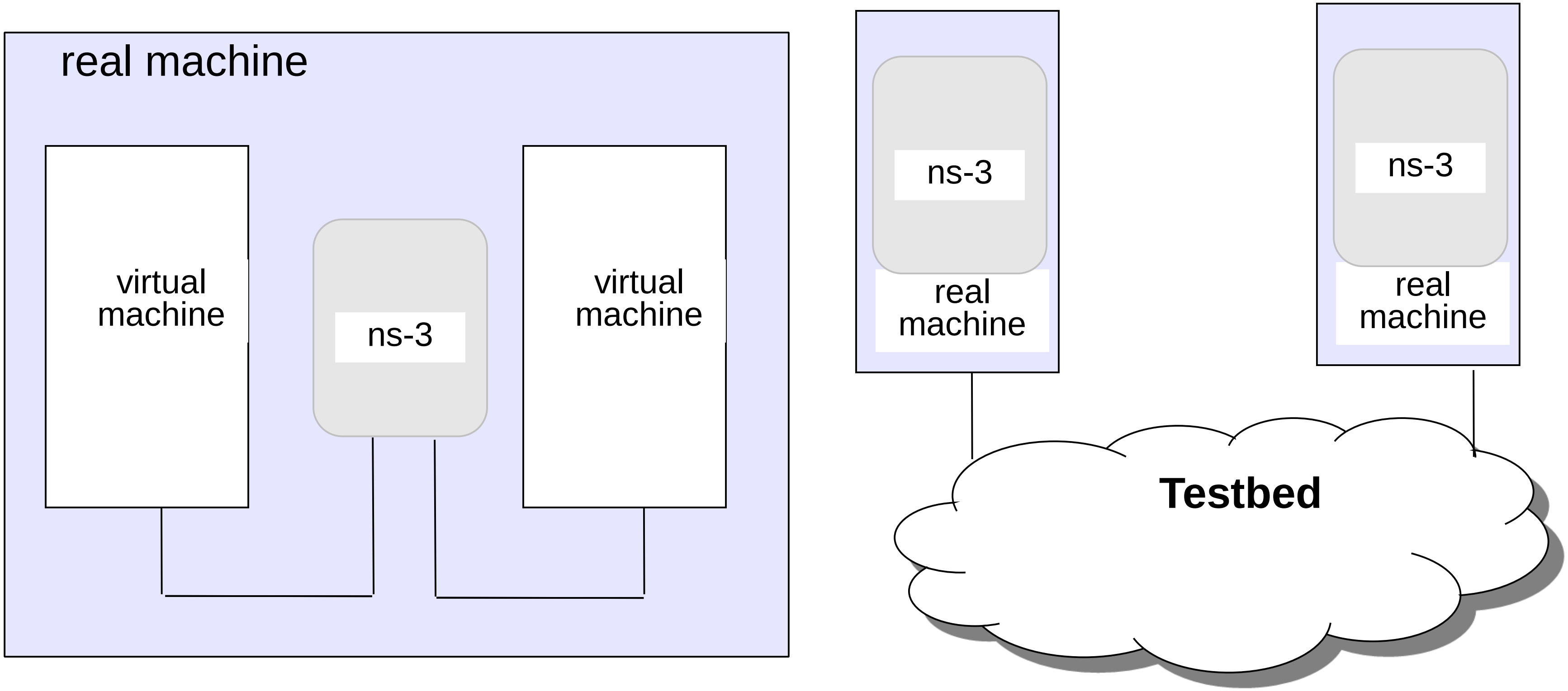


Direct Code Execution

- DCE/ns-3 framework requires the virtualization of a series of services
  - Multiple isolated instances of the same protocol on the same machine
- System calls are captured and treated by DCE
- Network stack protocols calls are captured and redirected
- To perform its work DCE re-implement the Linux program loader and parts of *libc* and *libpthread*



ns-3 emulation modes



1) ns-3 interconnects real or virtual machines

2) testbeds interconnect ns-3 stacks

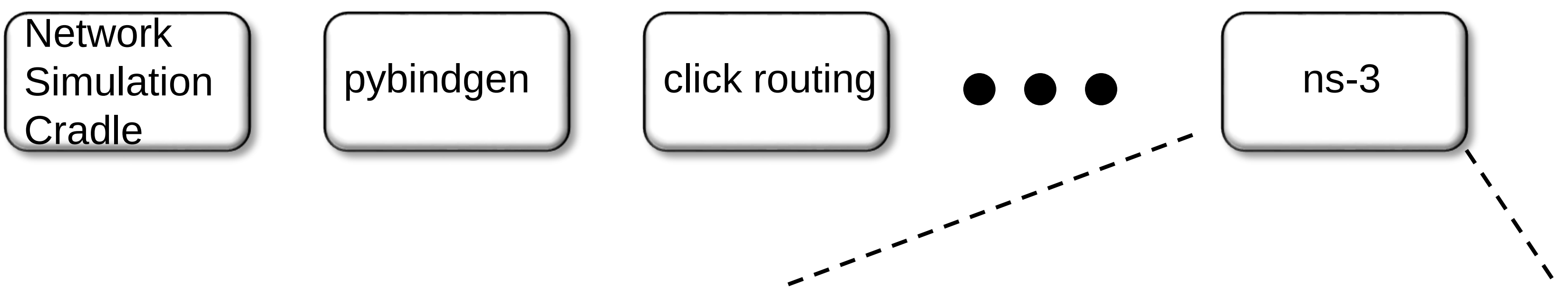
Various hybrids of the above are possible

ns-3 build systems

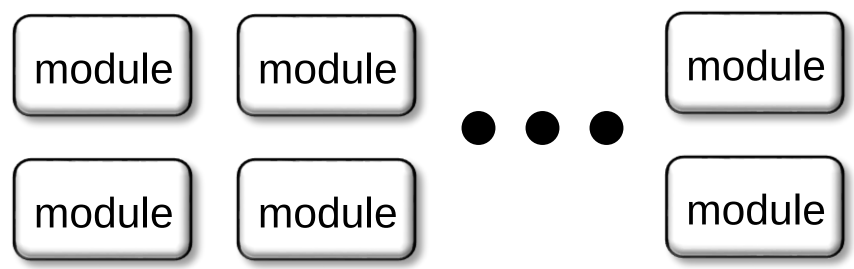
Software building

- Two levels of ns-3 build

1) bake (a Python-based build system to control an ordered build of ns-3 and its libraries)



2) waf, a build system written in Python



3) build.py (a custom Python build script to control an ordered build of ns-3 and its libraries) **<--- may eventually be deprecated**

ns-3 uses the 'waf' build system

- Waf is a Python-based framework for configuring, compiling and installing applications.
  - It is a replacement for other tools such as Autotools, Scons, CMake or Ant
  - http://code.google.com/p/waf/
- For those familiar with autotools:
- `configure` → `./waf configure`
- `make` → `./waf build`

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`

wscript example

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-

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
```

NS-3 NETWORK SIMULATOR

Build variations

- Configure 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

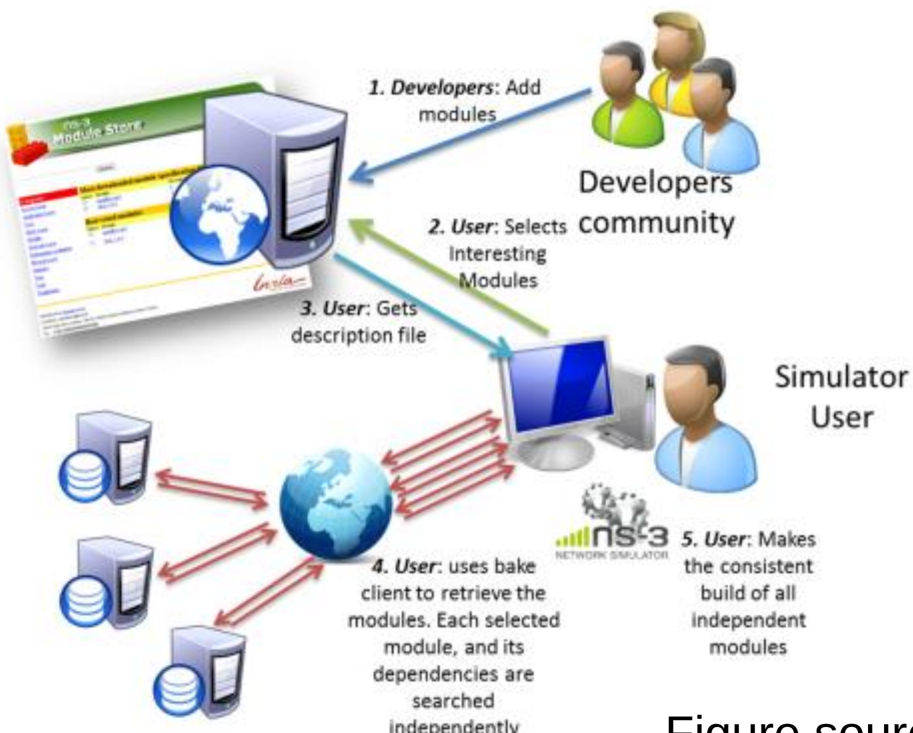
Building without wscript

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

Demo how programs can be built without wscripts

bake overview

- 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

Placeholder slide for demoing bake

Demo: `./waf build`

Look at: `build/` libraries and executables