

---

# ns-3 Training

**Session 1: Monday May 11**

**ns-3 Annual meeting  
May 2015**

# Introduction and logistics

---

- CTTC facilities
- Meals and coffee
- Wi-Fi
- Wiki page:
  - <https://www.nsnam.org/wiki/AnnualTraining2015>
- Meet your instructors

# Monday agenda

---

- Monday
  - ns-3 survey and overview tutorial, starting from first principles and walking through the running of simulations, configuration management, architecture of the software core, network emulation, and development practices using ns-3.
  - Methodology and workflow for developing new models in ns-3, using a case study.
  - Several tools used to extract and visualize data from ns-3 simulations, including the flow monitor, network animator NetAnim, Python-based visualizer, and the ns-3 tracing system.

# Tuesday agenda

---

- Tuesday
  - (09:00-10:30) Large-scale, distributed simulations with ns-3 (instructor: Peter Barnes)
  - (11:00-12:30) An introduction to the Direct Code Execution (DCE) environment, enabling users to use real application and Linux networking code in ns-3 (instructor: Hajime Tazaki)
  - Lunch break
  - (14:00-16:00) A survey of the LTE models, including model architecture, propagation models, LTE Radio Protocol Stack and EPC model. (instructor: Nicola Baldo)
  - (16:30-18:00) A tutorial on vehicular communication simulations, including mobility, WiFi and WAVE models, and propagation. (instructor: Konstantinos Katsaros)

# Later in the week

---

- WNS3 Wednesday and Thursday morning
- ns-3 Consortium Annual Meeting (16h00 Thursday)
- Developer meetings Friday

# 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

# Motivations for ns-3 project

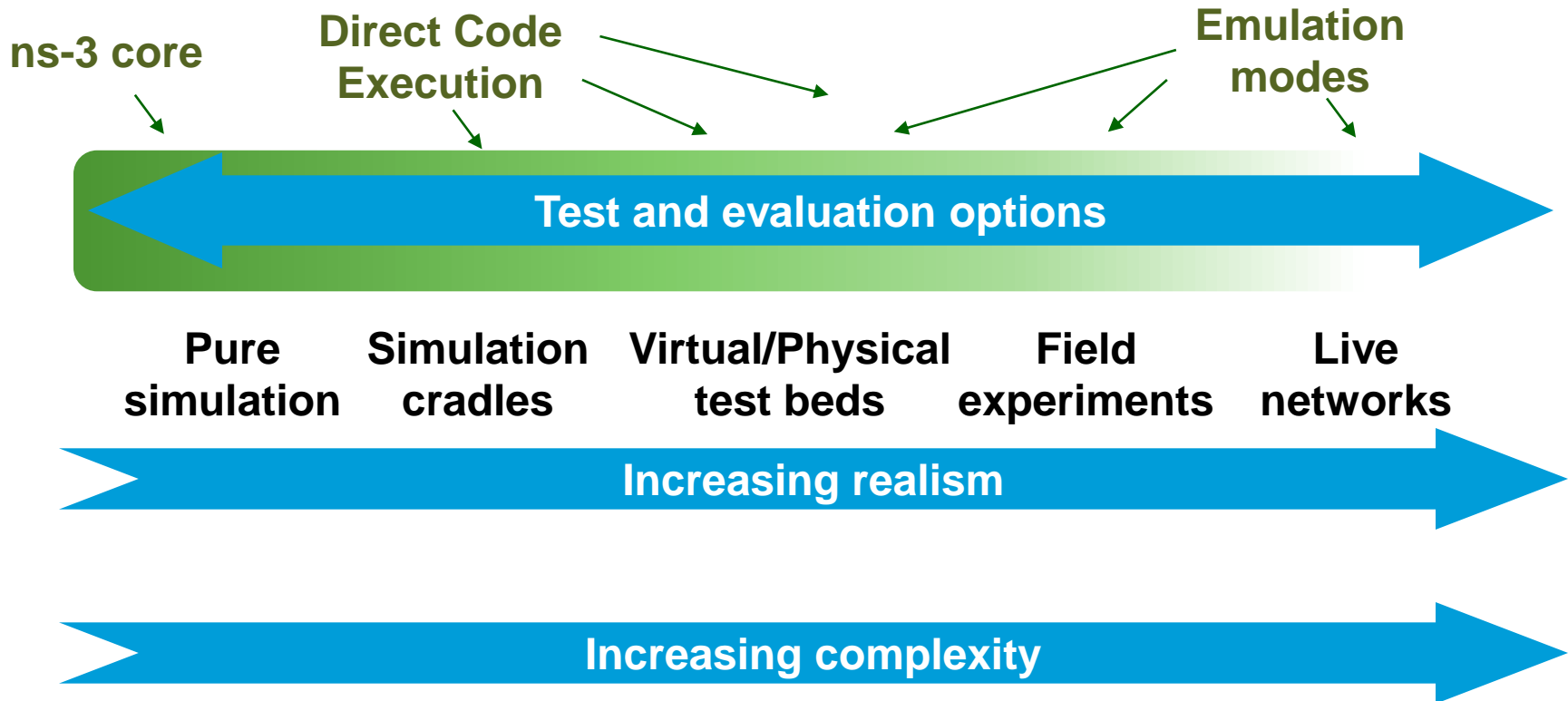
---

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

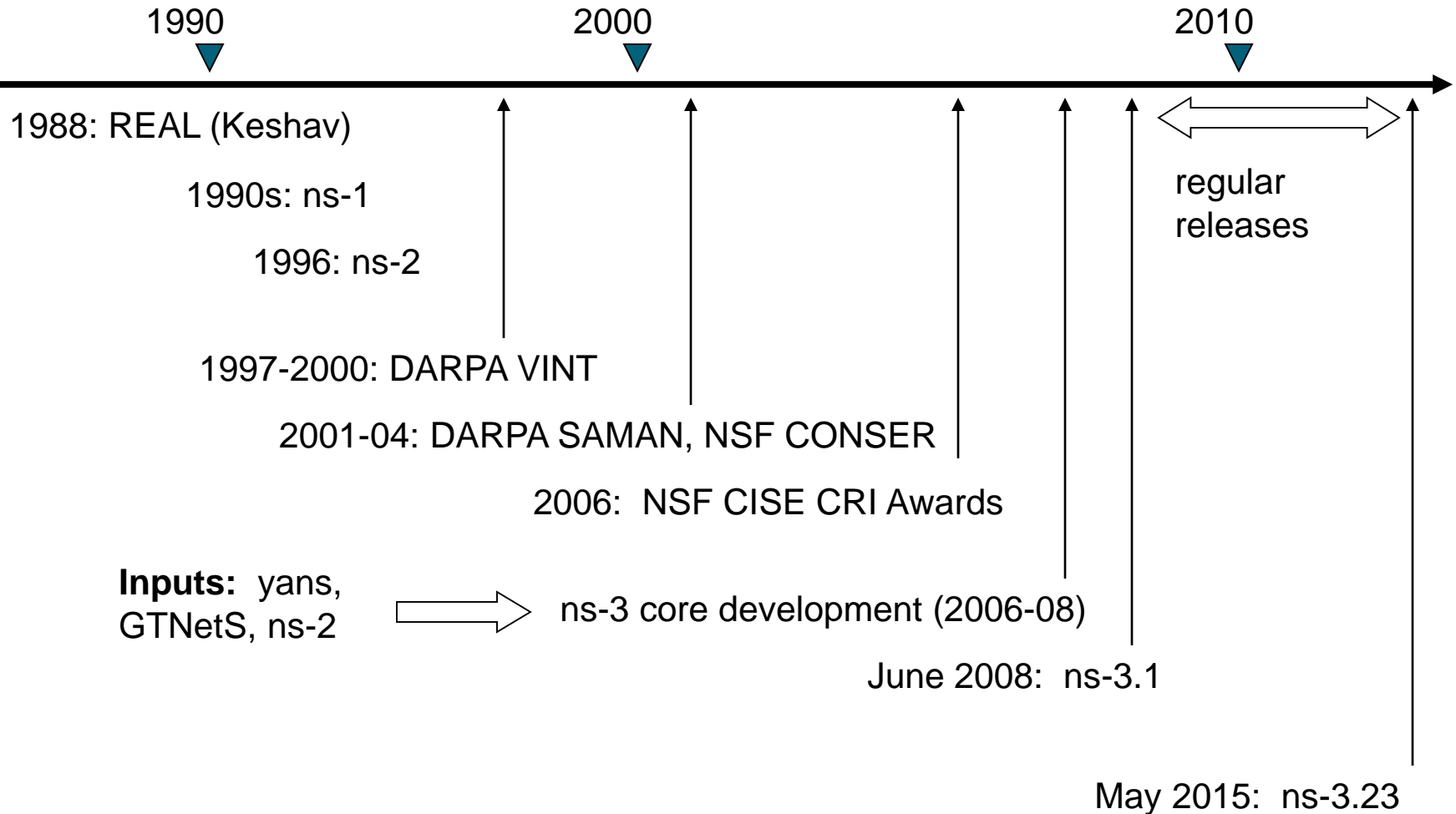
# Network performance evaluation options

- ns-3 enables researchers to more easily move between simulations, test beds, and experiments





# ns history



# 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

# 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.

# Publications using ns-3

---

A common question is "How many ns-3 papers are there?"

- Small survey of 139 paper results from 2013-14 search of IEEE library (top relevant results)
- Some papers matched multiple categories
- Hot topics:
  - LTE/cellular networks (15)
  - Wireless routing protocols (14)
  - Sensor networks (13)
  - Wireless MAC and PHY protocols (11)

# Paper counts by topic

Topic	Count	Topic	Count
LTE/Cellular	15	Network coding	4
Wireless routing protocols	14	Datacenter networks	4
Wireless sensor networks	13	Distributed systems	4
Wireless MAC/PHY	11	Optical links	3
Wireless QoS	9	Misc. physical links	3
Vehicular networks	9	Multicast	3
TCP/congestion control	9	Misc. security	2
Wireless security	9	Wired routers	2
About ns-3 itself	8	Wireless QoS	2
Wifi/mesh networks	7	WiMAX	1
Voice/video apps	6	Mobility	1
Energy/resource consumption	6	Misc. routing	1
DTN and space networks	5	Miscellaneous	1
Misc. wireless	5		

# Acknowledgment of support

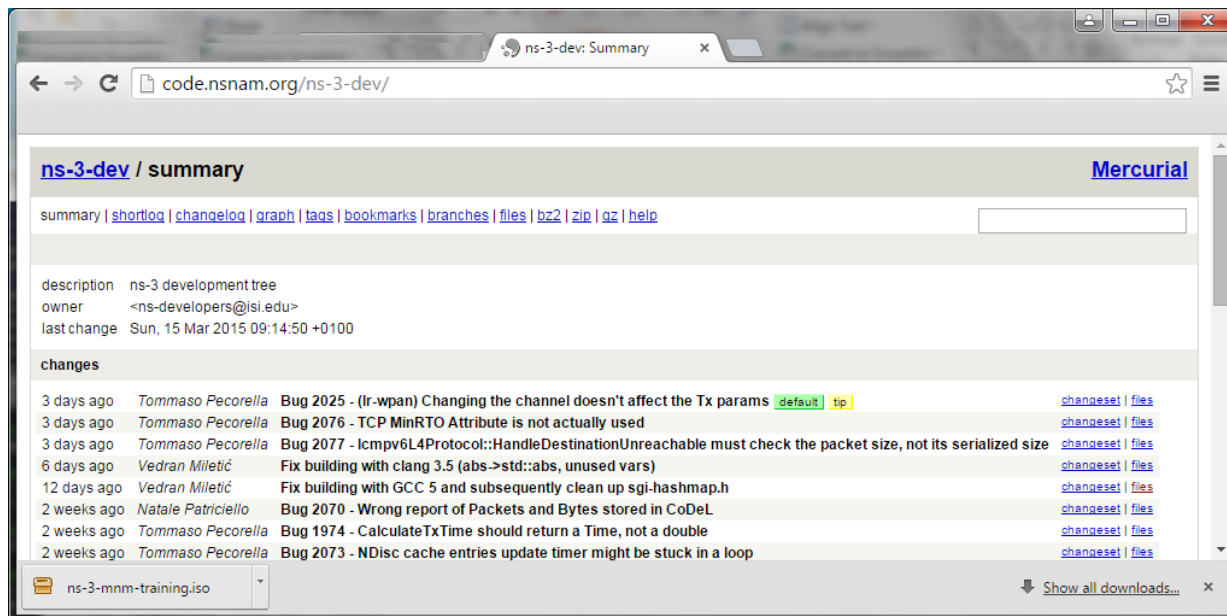


- 
- Software overview



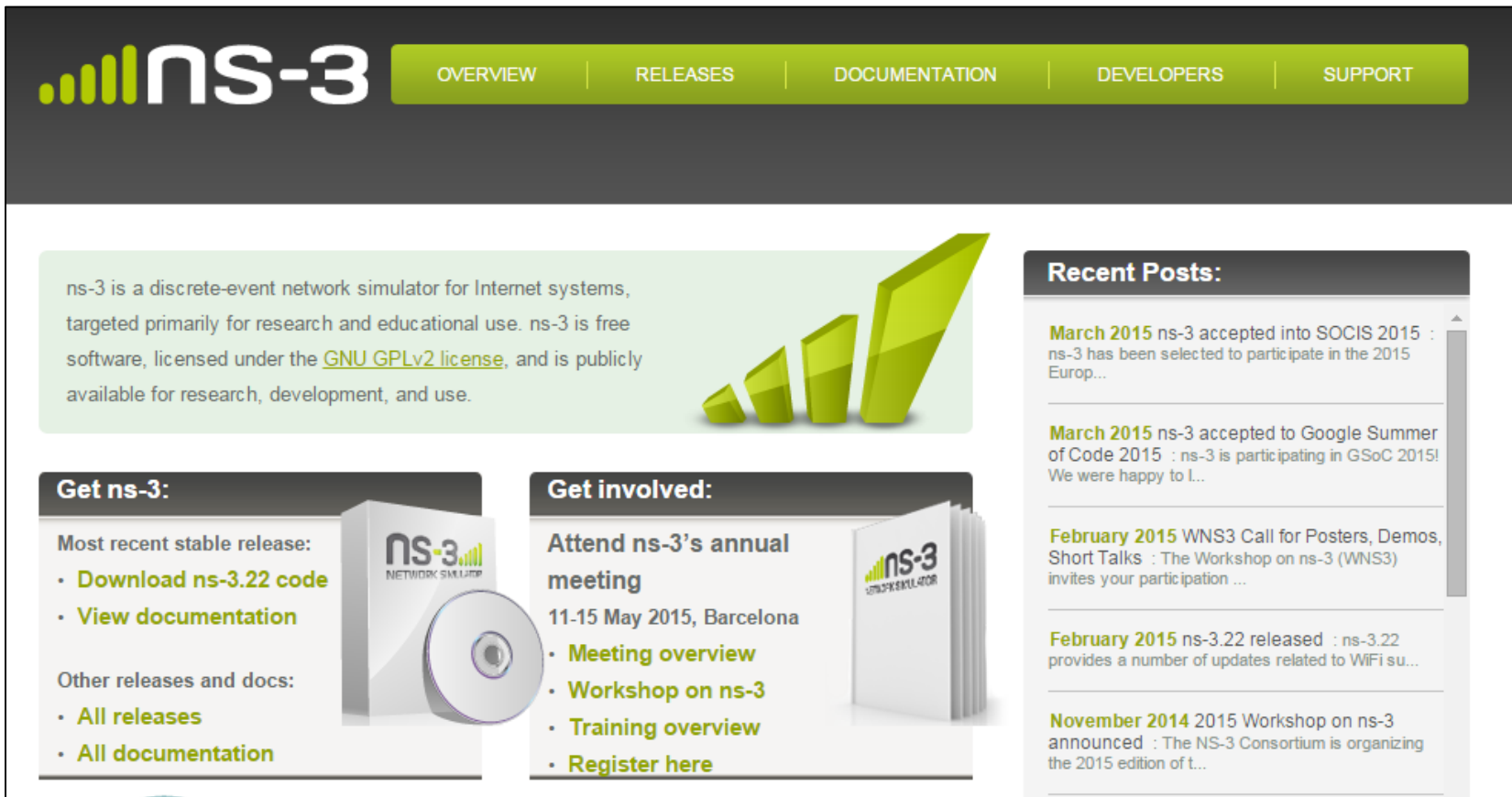
# Options for working along

- 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 main website

- Project home: <https://www.nsnam.org>



The screenshot shows the ns-3 main website homepage. At the top, there is a navigation bar with the ns-3 logo on the left and five menu items: OVERVIEW, RELEASES, DOCUMENTATION, DEVELOPERS, and SUPPORT. Below the navigation bar, there is a main content area. On the left, there is a text box describing ns-3 as a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. To the right of this text is a 3D bar chart with four bars of increasing height. Below the text box, there are two columns of content. The first column is titled 'Get ns-3:' and contains information about the most recent stable release (ns-3.22) and other releases and docs. The second column is titled 'Get involved:' and contains information about attending ns-3's annual meeting and participating in various activities. On the right side of the page, there is a 'Recent Posts:' section with a scrollable list of recent blog posts, including announcements about ns-3 being accepted into SOCIS 2015, Google Summer of Code 2015, WNS3 Call for Posters, Demos, Short Talks, and the release of ns-3.22.

ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the [GNU GPLv2 license](#), and is publicly available for research, development, and use.

**Get ns-3:**

Most recent stable release:

- [Download ns-3.22 code](#)
- [View documentation](#)

Other releases and docs:

- [All releases](#)
- [All documentation](#)

**Get involved:**

Attend ns-3's annual meeting

11-15 May 2015, Barcelona

- [Meeting overview](#)
- [Workshop on ns-3](#)
- [Training overview](#)
- [Register here](#)

**Recent Posts:**

**March 2015** ns-3 accepted into SOCIS 2015 : ns-3 has been selected to participate in the 2015 Europ...

**March 2015** ns-3 accepted to Google Summer of Code 2015 : ns-3 is participating in GSoc 2015! We were happy to L...

**February 2015** WNS3 Call for Posters, Demos, Short Talks : The Workshop on ns-3 (WNS3) invites your participation ...

**February 2015** ns-3.22 released : ns-3.22 provides a number of updates related to WiFi su...

**November 2014** 2015 Workshop on ns-3 announced : The NS-3 Consortium is organizing the 2015 edition of t...

# Software overview

---

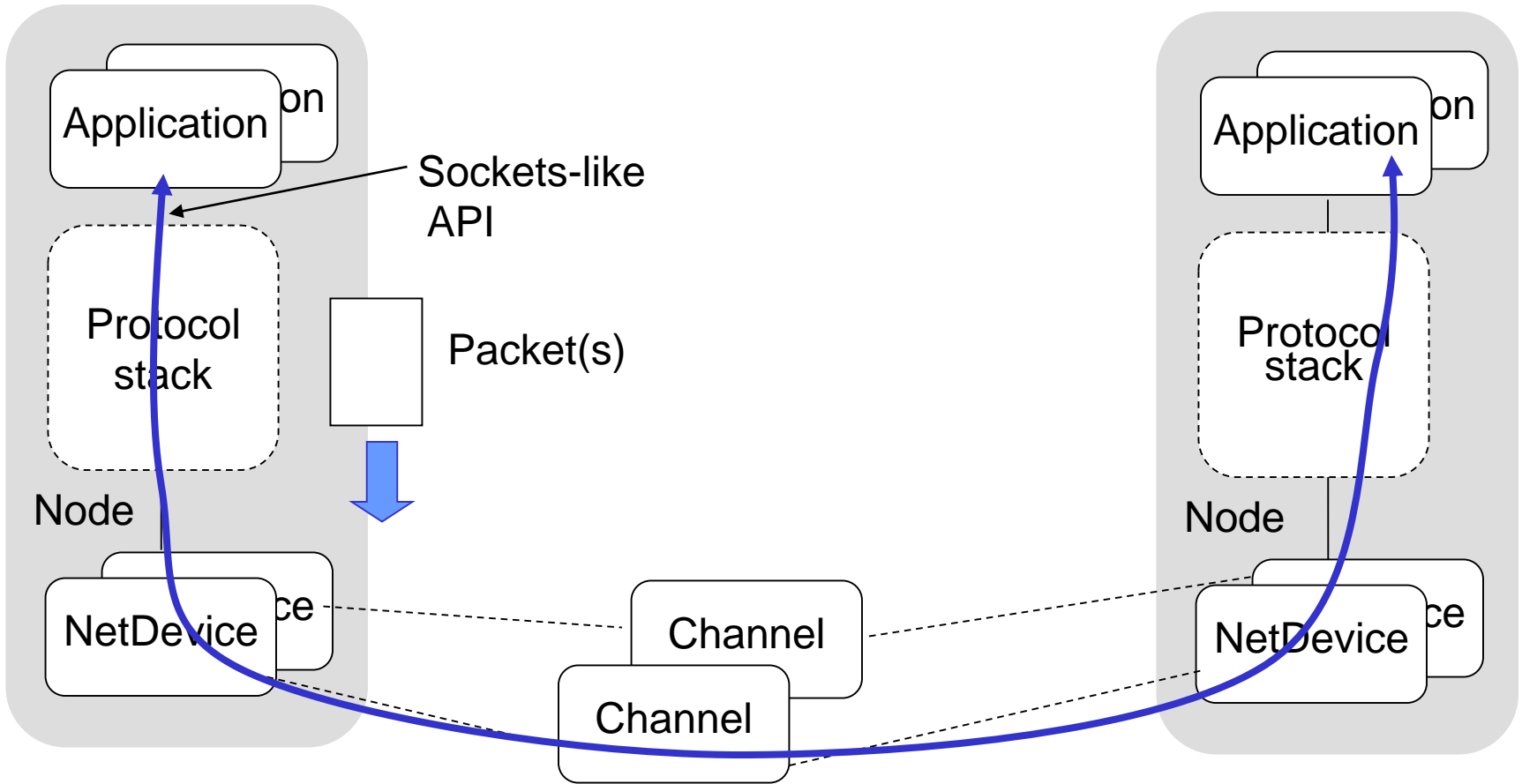
- ns-3 is written in C++, with bindings available for Python
  - simulation programs are C++ executables or Python programs
  - ~350,000 lines of C++ (estimate based on cloc source code analysis)
- ns-3 is a GNU GPLv2-licensed project
- ns-3 is mainly supported for Linux, OS X, and FreeBSD
  - Windows 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

# The basic ns-3 architecture



# Software orientation

---

Key differences from other network simulators:

1) Command-line, Unix orientation

– vs. Integrated Development Environment (IDE)

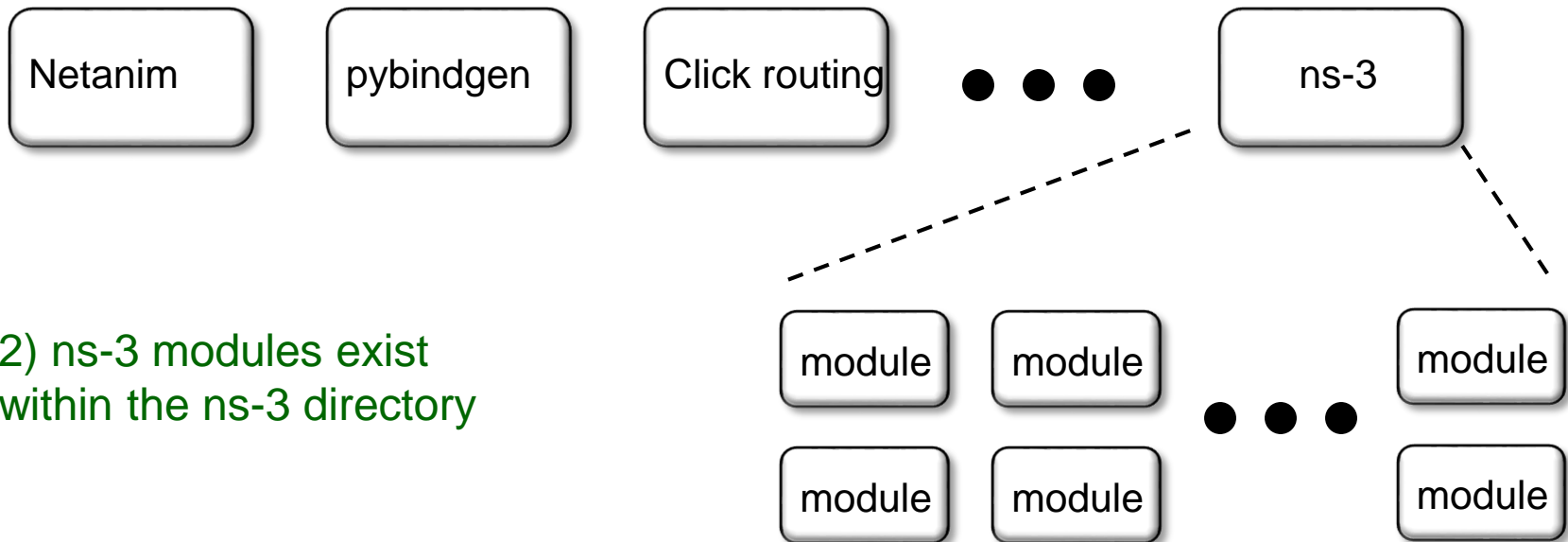
2) Simulations and models written directly in C++ and Python

– vs. a domain-specific simulation language

# Software organization

- Two levels of ns-3 software and libraries

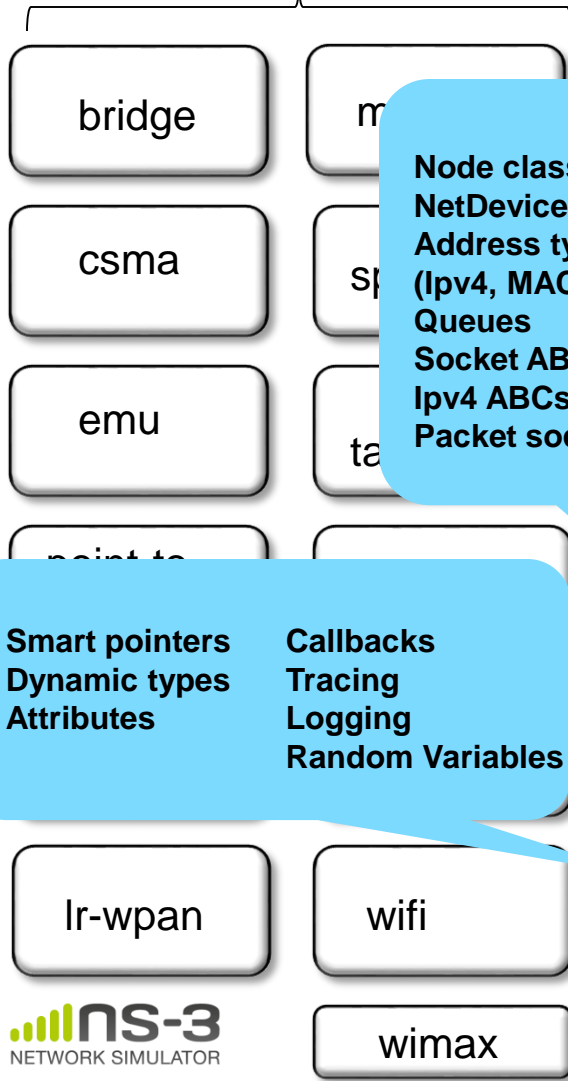
1) Several supporting libraries, not system-installed, can be in parallel to ns-3



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

# Current models

devices



Node class  
 NetDevice ABC  
 Address types  
 (Ipv4, MAC, etc.)  
 Queues  
 Socket ABC  
 Ipv4 ABCs  
 Packet sockets

Smart pointers  
 Dynamic types  
 Attributes

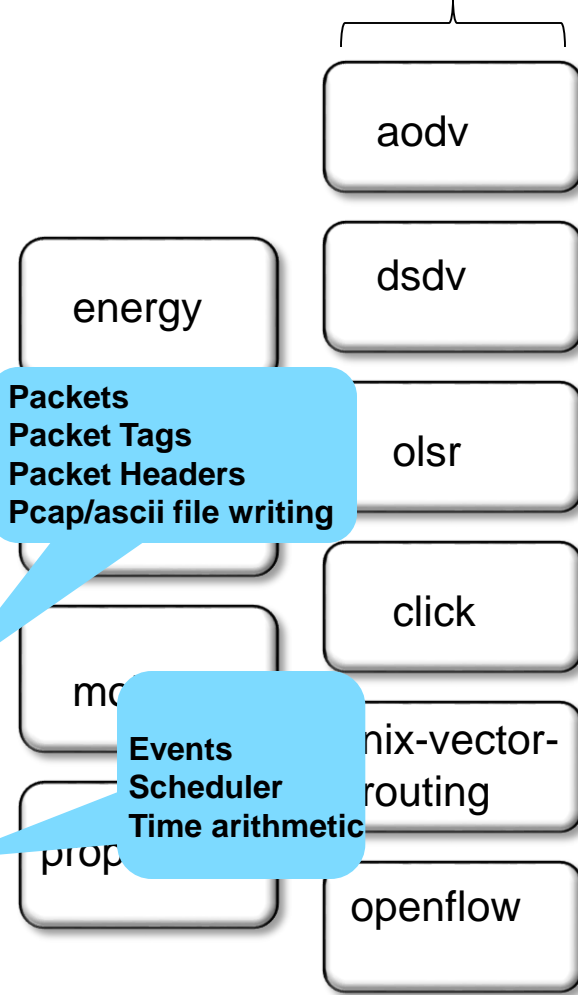
Callbacks  
 Tracing  
 Logging  
 Random Variables



Packets  
 Packet Tags  
 Packet Headers  
 Pcap/ascii file writing

Events  
 Scheduler  
 Time arithmetic

protocols



utilities





# Module organization

---

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

# ns-3 programs

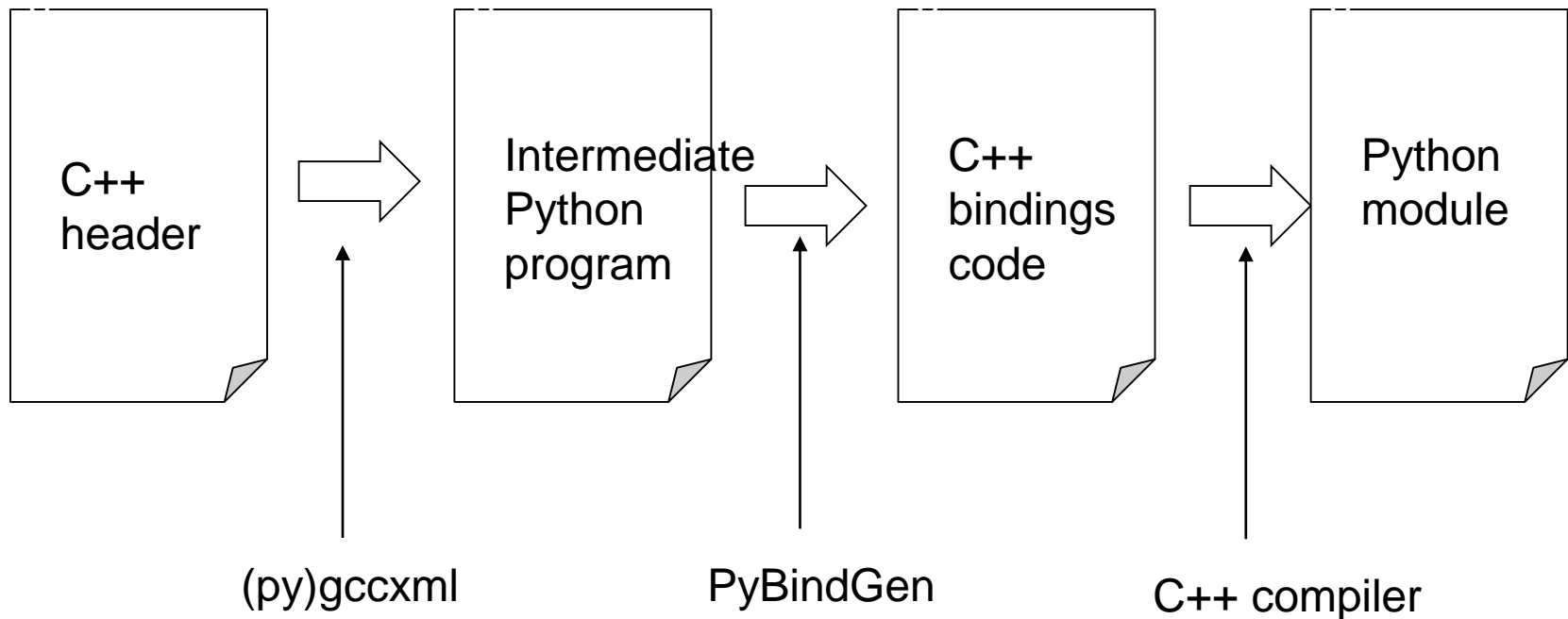
---

- 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`

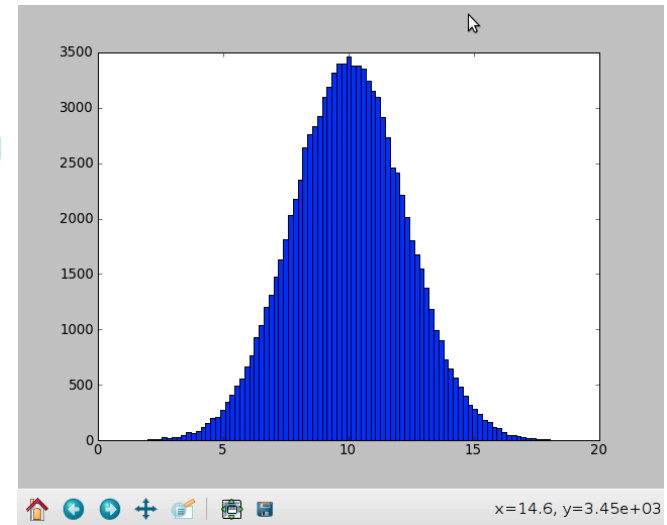
```
# Demonstrate use of ns-3 as a random number generator integrated  
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial
```

```
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)]
```

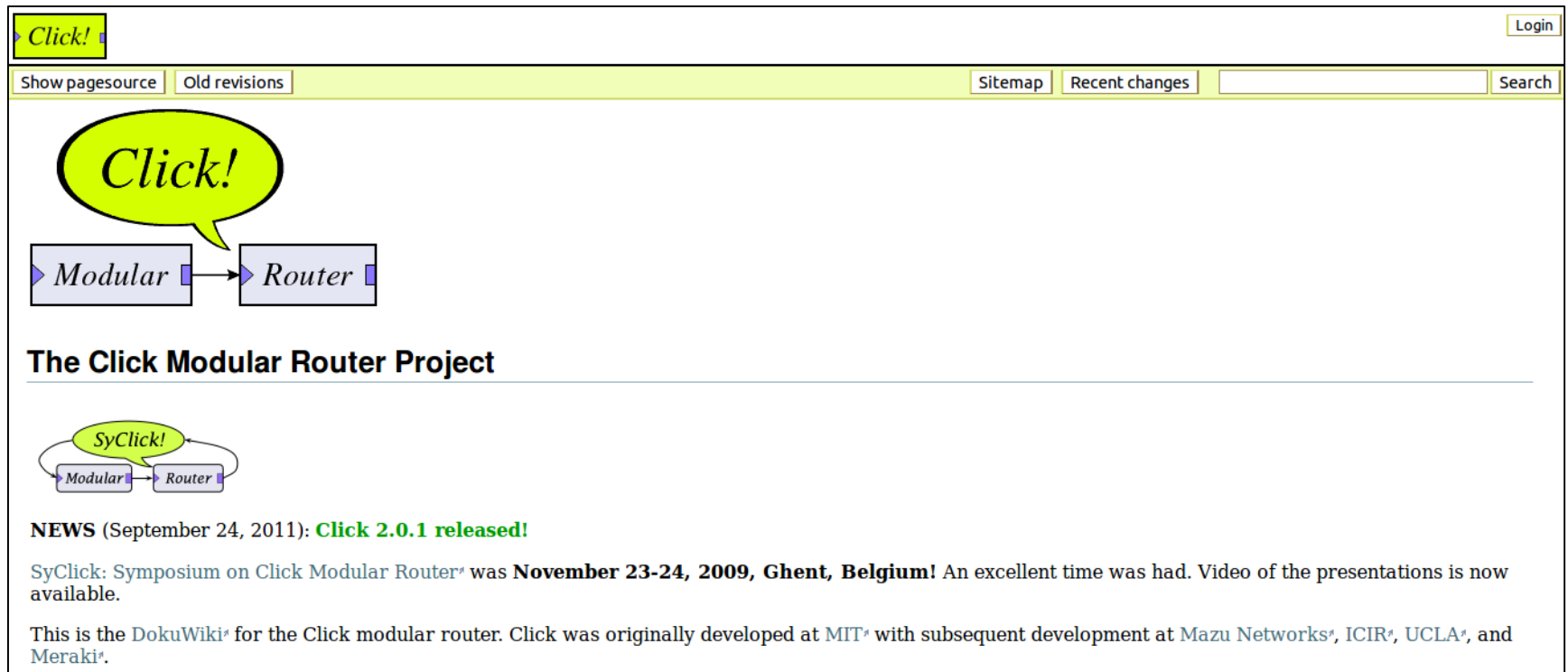
```
# the histogram of the data  
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)
```

```
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



The screenshot shows a web browser window with the following elements:

- Top left: A yellow button labeled "Click!".
- Top right: A "Login" button.
- Navigation bar: "Show pagesource", "Old revisions", "Sitemap", "Recent changes", and a "Search" input field.
- Main content area:
  - A yellow speech bubble containing the word "Click!".
  - A diagram showing a box labeled "Modular" with an arrow pointing to a box labeled "Router".
  - A section header: "The Click Modular Router Project".
  - A smaller diagram showing a box labeled "Modular" with an arrow pointing to a box labeled "Router", with a yellow speech bubble labeled "SyClick!" above it.
  - A "NEWS" section dated September 24, 2011, with the text: "Click 2.0.1 released!".
  - A paragraph: "SyClick: Symposium on Click Modular Router<sup>\*</sup> was **November 23-24, 2009, Ghent, Belgium!** An excellent time was had. Video of the presentations is now available.
  - A paragraph: "This is the DokuWiki<sup>\*</sup> for the Click modular router. Click was originally developed at MIT<sup>\*</sup> with subsequent development at Mazu Networks<sup>\*</sup>, ICIR<sup>\*</sup>, UCLA<sup>\*</sup>, and Meraki<sup>\*</sup>."

# mininet emulator

The screenshot shows the GitHub repository page for `mininet/mininet`. The repository is public and has 468 stars and 204 forks. The current page is titled "Link modeling using ns 3". The page content includes a "Contents" section with a list of links:

- [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](#)

On the right side of the page, there is a sidebar with a table of contents:

- [Mininet](#)
- [Get Started](#)
- [Sample Workflow](#)
- [Walkthrough](#)
- [Overview](#)

Below this, there is another list of links:

- [Download](#)
- [Documentation](#)
- [Videos](#)
- [Source Code](#)
- [Apps](#)
- [FAQ](#)
- [Wiki](#)
- [Teaching](#)
- [Papers](#)
- [GSoC 2013](#)

# Co-simulation frameworks have emerged

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

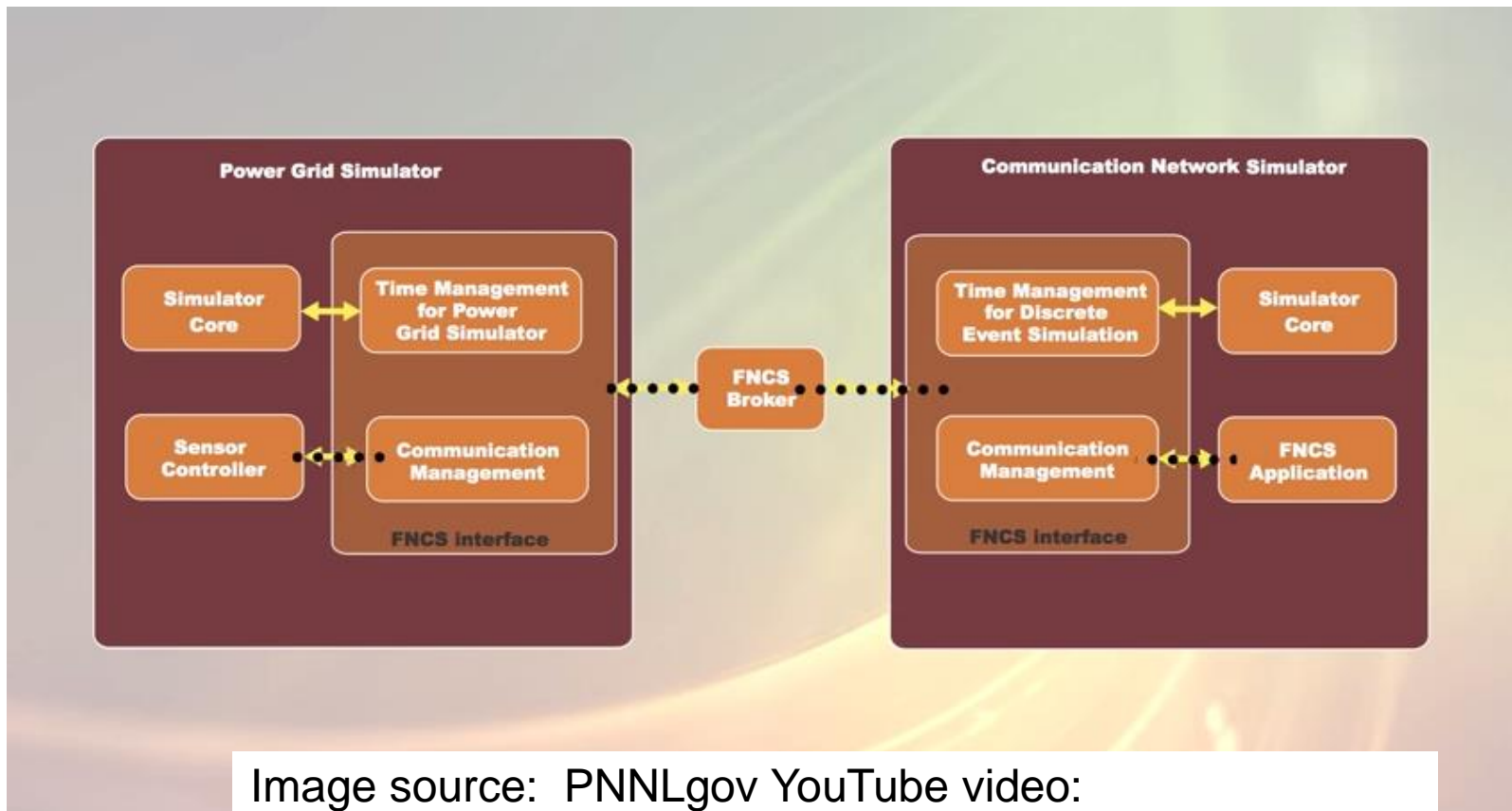


Image source: PNNLgov YouTube video:  
Introducing FNCS: Framework for Network Co-Simulation

# FAQs

---

- Does ns-3 have a Windows version?
  - Yes, for Visual Studio 2012
  - [http://www.nsnam.org/wiki/Ns-3\\_on\\_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](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)?
  - Ubuntu/Debian packages for ns-3.17 release
- 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>

The screenshot shows a web browser window displaying the ns-3 Doxygen API documentation for the `ArpCache` class. The browser's address bar shows the URL `https://www.nsnam.org/doxygen/classes3_1_1_arp_cache.html#details`. The page header includes the ns-3 logo and navigation links for HOME, TUTORIALS, DOCS, and DEVELOP. The main navigation bar has tabs for Main Page, Related Pages, Modules, Namespaces, Classes, and Files. The left sidebar shows a class hierarchy with `ArpCache` selected. The main content area is titled "Detailed Description" and contains the following text:

An ARP cache.

A cached lookup table for translating layer 3 addresses to layer 2. This implementation does lookups from IPv4 to a MAC address

**Config Paths**

ns3::ArpCache is accessible through the following paths with **Config::Set** and **Config::Connect**:

- `"/NodeList/[i]/$ns3::Ipv4L3Protocol/InterfaceList/[i]/ArpCache"`
- `"/NodeList/[i]/$ns3::ArpL3Protocol/CacheList/[i]"`

**Attributes**

Generated on Tue Mar 17 2015 18:22:57 for ns-3 by doxygen 1.8.9.1

# Contributed code and associated projects

Overall ndnSIM documentation

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
  - Stackbuilder
    - Routing
      - Manually routes
      - Global routing controller
      - Default routes
    - Content Store
    - Pending Interest Table
    - Forwarding strategy
  - Appletiger
  - 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 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-RT0 algorithms

### Getting Started

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

KIT  
Karlsruhe Institute of Technology

Decentralized Systems and Network Services Research Group - TM & SCC

### PhysSimWiFi for NS-3

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

### Overview

PhysSim-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 packet and the length of the packet, the PhysSim-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 incorporate 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 PhysSim-WiFi implementation is a drop-in replacement of the default YansWiFPHY 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

- [PhysSimWiFi Manual 1.1](#)

Full package download

- [NS-3.9-PhysSimWiFi v1.1](#) (based on NS-3.9 and PhysSim-WiFi) - August 19, 2011
- [NS-3.9-PhysSimWiFi v1.0](#) (based on NS-3.9 and PhysSim-WiFi) - September 12, 2010

Patches for NS-3

- [PhysSimWiFi v1.1 for NS-3.9](#) - August 19, 2011
- [PhysSimWiFi v1.1 for NS-3.9-PhysSimWiFi v1.0](#) - August 19, 2011
- [PhysSimWiFi v1.0 for NS-3.9](#) - September 12, 2010

# 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

---

# ns-3 build systems

# 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`

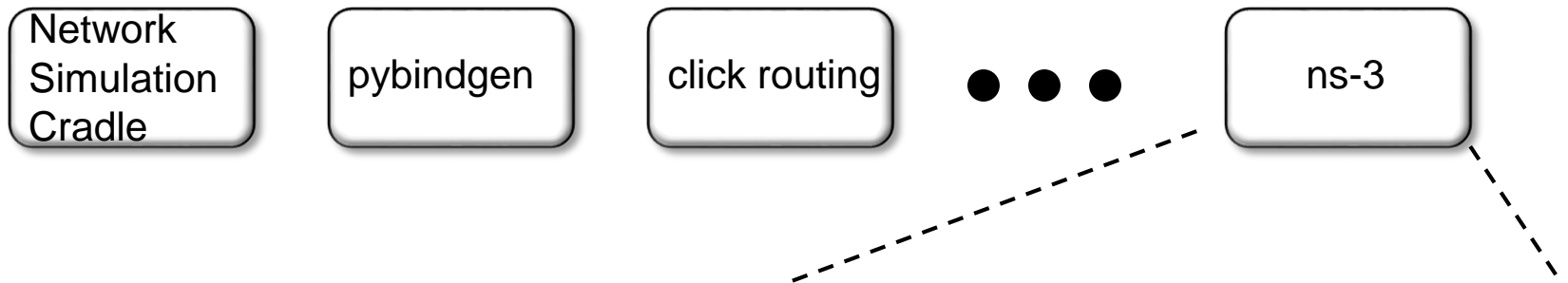
Q. What is "**hg clone**"?

A. Mercurial (<http://www.selenic.com>) is our source code control tool.

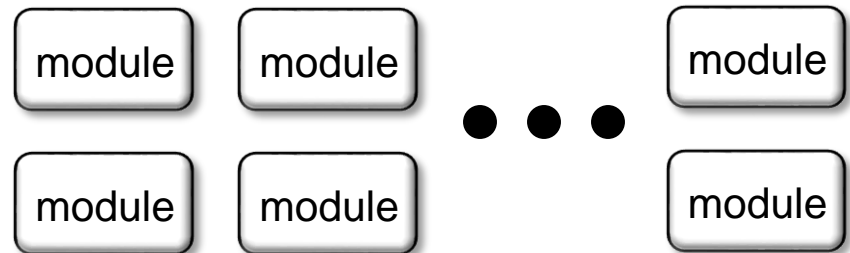
# Software building

- 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
```

# 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

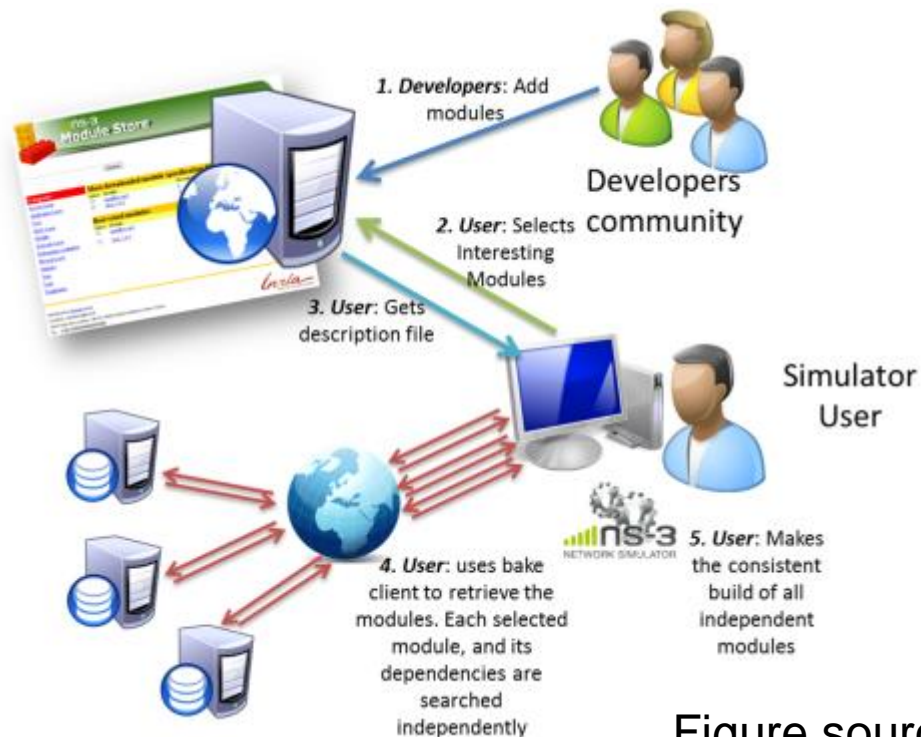
---

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

Demo how programs can be built without wscripts

# 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

# bake basics

---

- bake can be used to build the Python bindings toolchain, Direct Code Execution, Network Simulation Cradle, etc.
- Manual available at <https://www.nsnam.org/docs/bake/tutorial/html/index.html>

```
./bake.py configure -e <module>
```

```
./bake.py show
```

```
./bake.py download
```

```
./bake.py build
```



---

# Placeholder slide for demoing bake

Demo: `./waf build`

Look at: `build/` libraries and executables

---

# Visualization

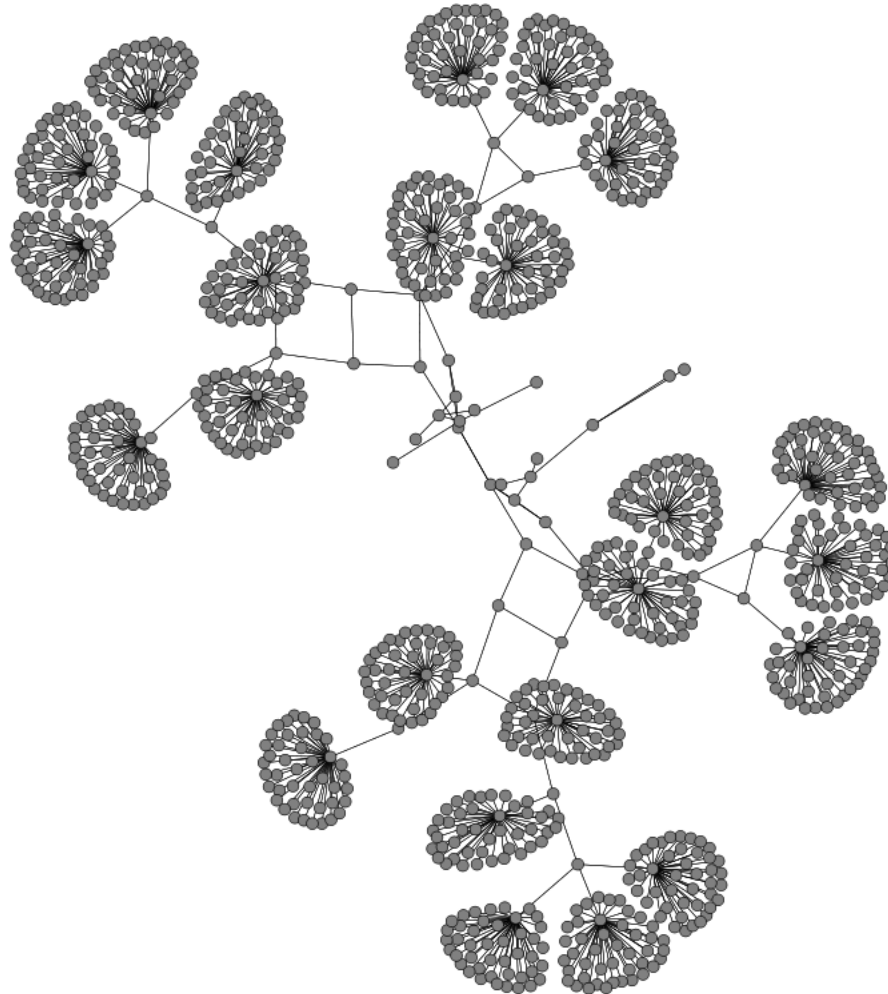
# PyViz overview

---

- 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

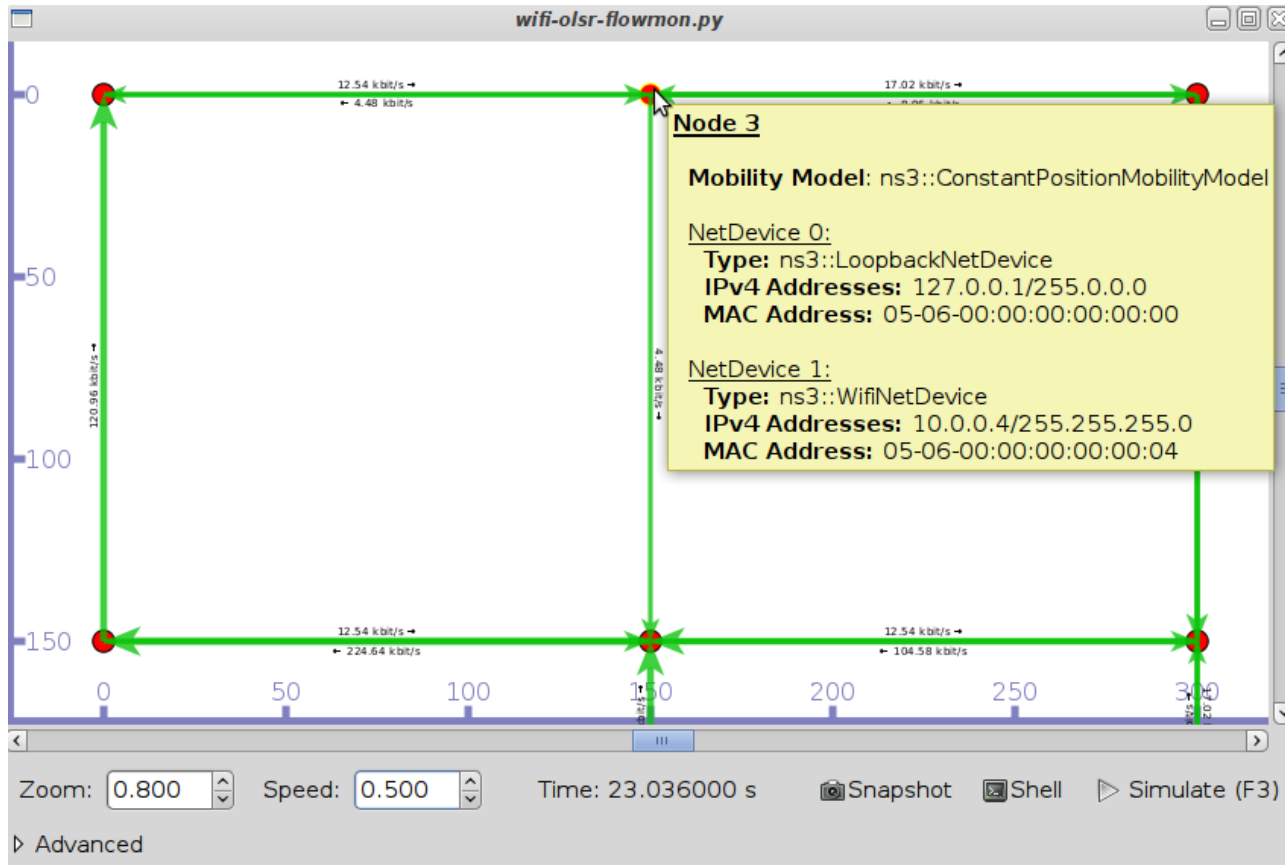
# Pyviz screenshot (Graphviz layout)

---



# 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

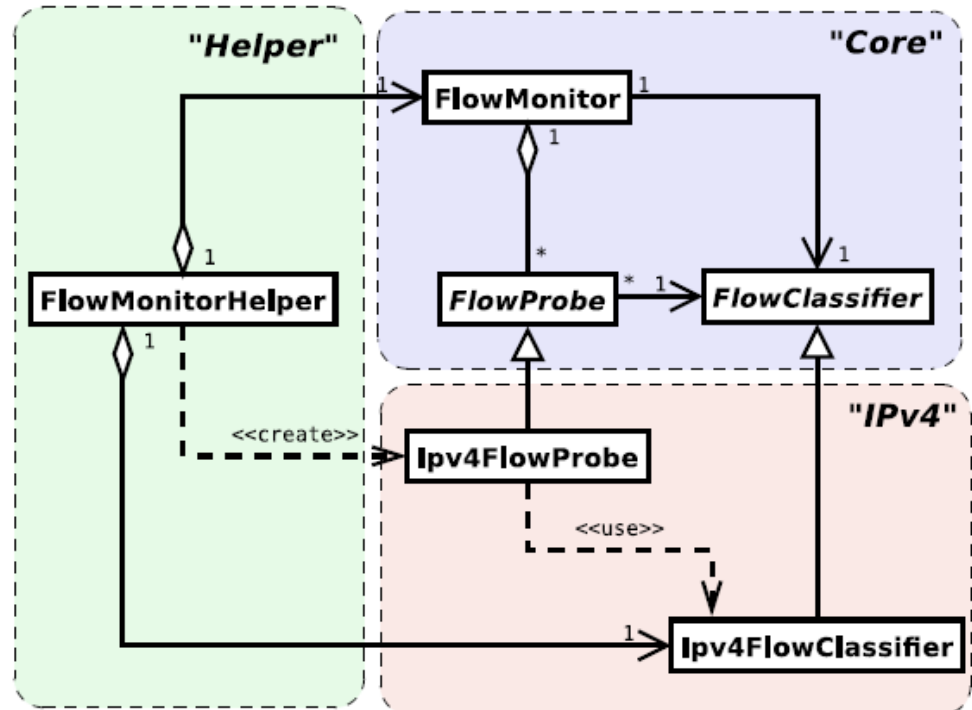
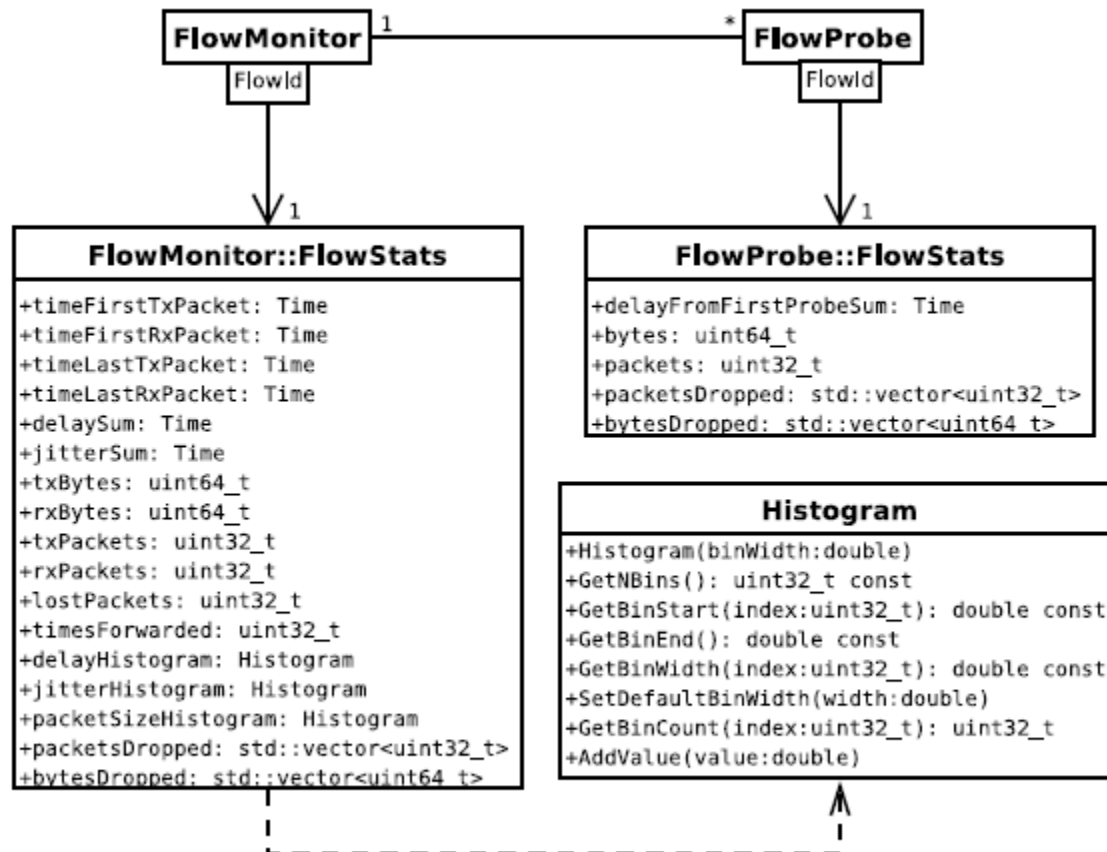


Figure credit: G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.



# 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

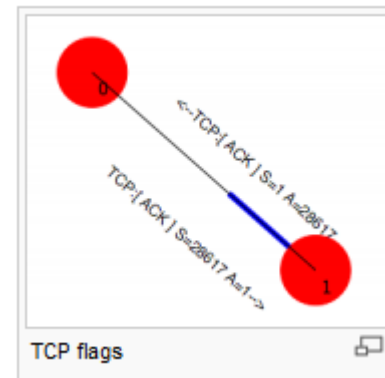
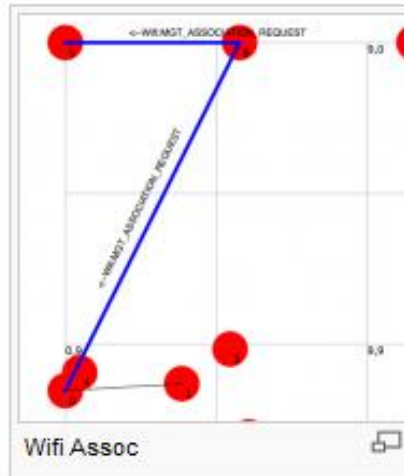
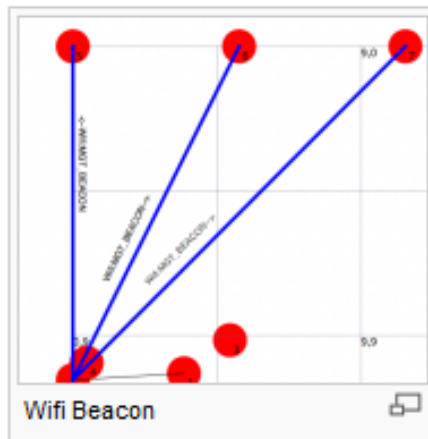
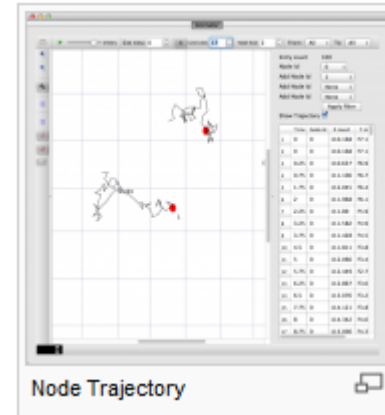
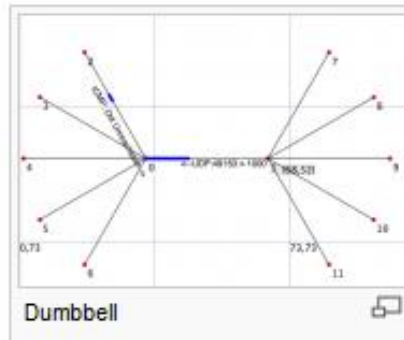
```
Hidden station experiment with RTS/CTS disabled:  
Flow 1 (10.0.0.1 -> 10.0.0.2)  
  Tx Bytes:   3847500  
  Rx Bytes:   316464  
  Throughput: 0.241443 Mbps  
Flow 2 (10.0.0.3 -> 10.0.0.2)  
  Tx Bytes:   3848412  
  Rx Bytes:   336756  
  Throughput: 0.256924 Mbps  
-----  
Hidden station experiment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

No	Time	From Node Id	To Node Id	Packet
1	2.5e-05	0	5	WIFI_MGT_BEACON FromDS: 0 ToDS: 0 DA: 8:8:8:8:8:8
2	2.3e-05	0	6	WIFI_MGT_BEACON FromDS: 0 ToDS: 0 DA: 8:8:8:8:8:8
3	2.5e-05	0	7	WIFI_MGT_BEACON FromDS: 0 ToDS: 0 DA: 8:8:8:8:8:8
4	0.000167003	5	0	WIFI_MGT_ASSOCIATION_REQUEST FromDS: 0 ToDS: 1
5	0.000167003	5	7	WIFI_MGT_ASSOCIATION_REQUEST FromDS: 0 ToDS: 1
6	0.000167003	5	0	WIFI_MGT_ASSOCIATION_REQUEST FromDS: 0 ToDS: 1
7	0.000179066	0	5	WIFI_CTL_ACK RA:80:80:80:80:80:07
8	0.000179066	0	6	WIFI_CTL_ACK RA:80:80:80:80:80:07
9	0.000179066	0	7	WIFI_CTL_ACK RA:80:80:80:80:80:07
10	0.000492183	6	5	WIFI_MGT_ASSOCIATION_REQUEST FromDS: 0 ToDS: 1
11	0.000492183	6	0	WIFI_MGT_ASSOCIATION_REQUEST FromDS: 0 ToDS: 1
12	0.00051414	0	5	WIFI_CTL_ACK RA:80:80:80:80:80:08
13	0.00051414	0	6	WIFI_CTL_ACK RA:80:80:80:80:80:08
14	0.00051414	0	7	WIFI_CTL_ACK RA:80:80:80:80:80:08

Packet Statistics



# 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

---