
ns-3 Training

**ns-3 Annual Meeting
June 2017**

ns-3 training goals

- Make attendees more productive with ns-3
 - 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

Agenda and Instructors

- Software and usage overview (T. Henderson)
- How to write new models (T. Pecorella)
- Wi-Fi and wireless models (T. Henderson)
- TCP and AQM models (M. Tahiliani)
- Traffic control (S. Avallone)

Please ask questions along the way!

Additional training archives

- LTE (Lorenza Giupponi and Biljana Bojovic), June 2016
- Parallel, Distributed Simulations (Peter Barnes), June 2016
- Direct Code Execution (Tom Henderson), June 2016
- Tracing (Walid Younes), June 2014

<http://www.nsnam.org/wiki/Training2017>

Your feedback on requested topics

1) what is your past level of experience with ns-3?

- various (from starting the tutorial to having written new models)

2) what technical topics in the simulator interest you the most?

- Wi-Fi, LTE, TCP
- routing 6LoWPAN, IoT, IPv6, BGP, and the core

3) past level of experience with any other network simulation tools?

- MATLAB/Simulink, plus ns-2, OPNET, OMNeT++, Totem

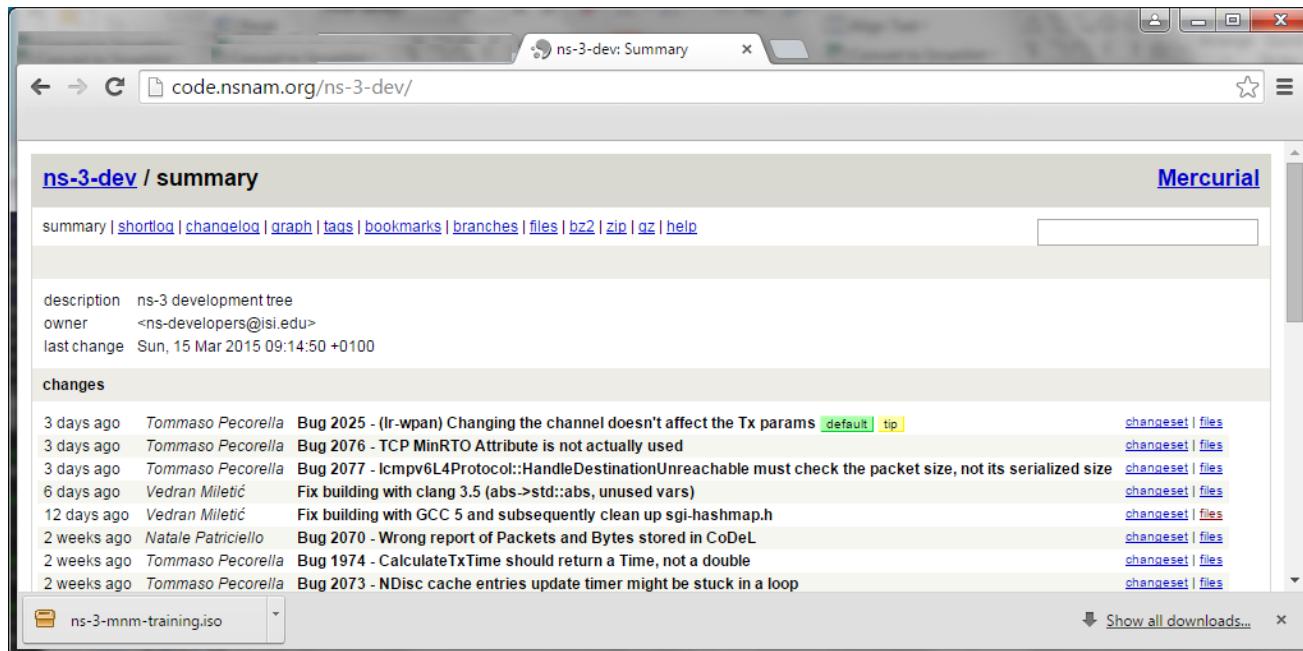
Your feedback on requested topics

4) what do you most want to get out of the training sessions

- refresh, get ideas for lab assignments, understand real-time simulations, inject real traffic, global tips and tricks about ns-3, learn LTE, implement new models

Options for working along

- 1) Download the required packages onto your (Linux, OS X, or BSD) system
- 2) Download the ISO image (Live DVD)
- 3) Browse the code online: <https://code.nsnam.org>



Project overview

Motivations for ns-3 project

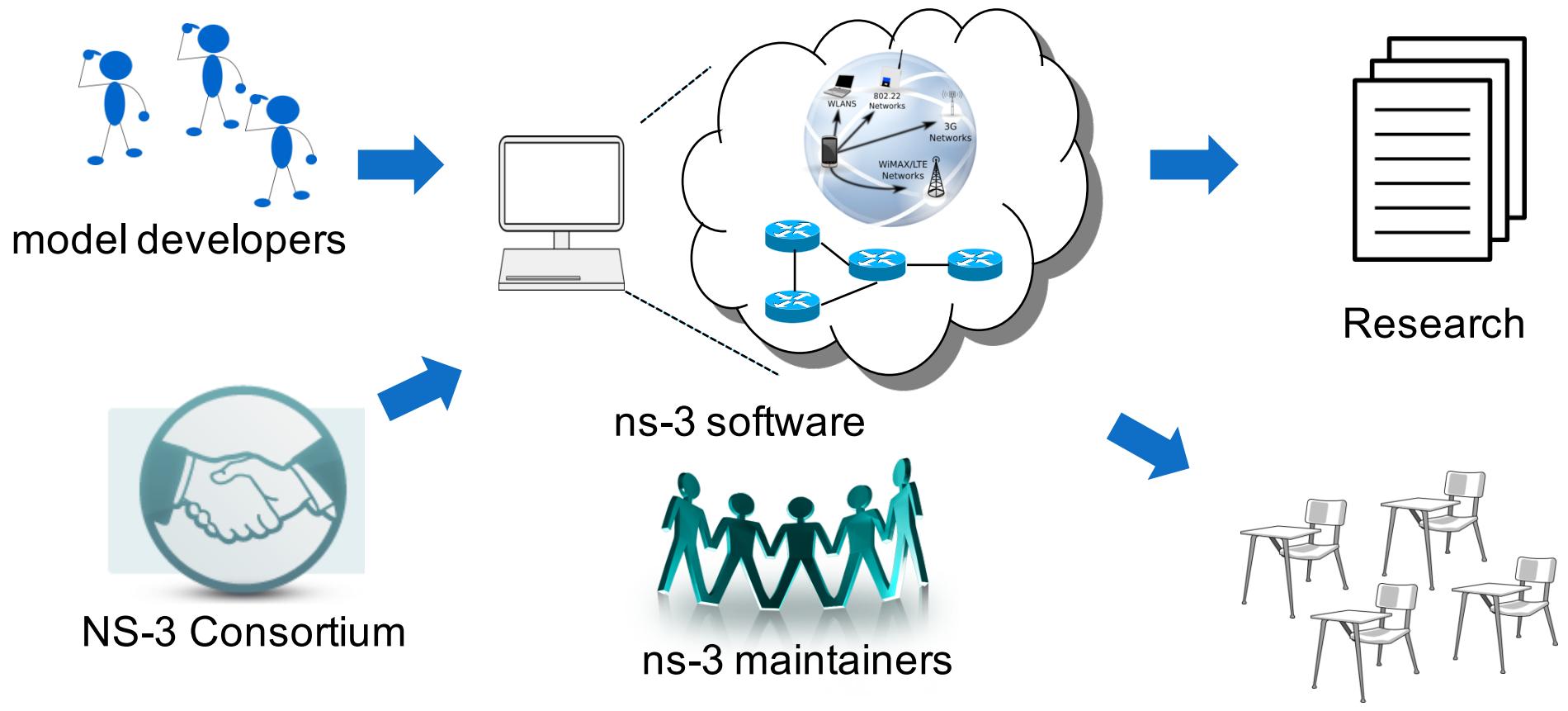
Develop an extensible simulation environment for networking research

- 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

Community-maintained, scientific computing software by following best current practices for open source

ns-3: An Open Source Network Simulator

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



What have people done with ns-3?

- thousands of publications to date
 - search of 'ns-3 simulator' on IEEE and ACM digital libraries, or Google Scholar

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

FSR: Formal Analysis and Implementation Toolkit for Safe Interdomain Routing

Anand Wang, Linlin Jia, Member, IEEE, Wenchao Zhou, Yiqing Ren, Boon Thau Loo, Jennifer Rexford, Senior Member, IEEE, Vivik Nigam, Andre Scedrov, and Carolyn Talcott

Abstract—Interdomain routing stitches the disparate parts of the Internet together using protocol stability as a critical measure to both researchers and operators. Researchers create safety proofs and counterexamples by hand and build simulators and prototypes to explore protocol dynamics. Similarly, network operators use specific configuration tools to manage using heterogeneous tools. In this paper, we present a comprehensive toolkit for analyzing and implementing router policies, ranging from hand-coded proofs to specific configurations. Our *Formally Safe Routing* (*FSR*) toolkit performs all of these functions in a formal setting. We demonstrate its usefulness by showing that routing algebra has a natural translation to both safety constraints (to perform safety analysis with SMF tools) and deriveable network-wide properties (to prove convergence). Our extensive experiments with realistic topologies and policies show how *FSR* can detect problems in an autonomous system's (AS) configuration, automatically generate conditions for Router Configuration Safety (RCF) safety, and empirically evaluate convergence time.

Index Terms—Communications technology, declarative networking, formal analysis, routing algebra.

1. INTRODUCTION

THE INTERNET's global routing system does not necessarily converge, depending on the how the Border Gateway Protocol (BGP) policies and individual routers are configured. When two overlapping areas, sometimes called disruptions and renege overhead, researchers devote significant attention to BGP stability (or "safety"). Abstract formal models of BGP [12]–[15], [16] allow researchers to explore how local policies affect BGP stability. Recently, researchers have proposed new guidelines for building a more robust Internet [22] to produce provably correct implementations of safe interdomain routing.

Given policy configurations as input, *FSR* produces an abstract safety properties and a distributed protocol implementation, as shown in Fig. 1. *FSR* has three main underlying technologies.

Policy configuration: Our first step is to generate policy configurations [13] [30] for providers and network operators to express policy configurations in an abstract algebraic form. These configurations can be anything from high-level policy *guidelines* (e.g., proposed constraints that a certain wireless link or a completely specified policy) to an *IS-IS* [23] or *inter-domain autonomous-system (AS)* network that an operator wants to analyze. Router configuration files can be automatically translated into the algebraic representation, easing the adoption of *FSR*.

Safety analysis: To automatically analyze the policy configuration, *FSR* reduces the convergence proof to a

Manuscript received May 23, 2011; accepted January 21, 2012; approved by the Communications Policy Committee. This work was supported in part by the National Science Foundation (NSF) under Grants CCF-0826024, CNS-0835949, CNS-0964683, and CNS-1053512; by the U.S. Air Force Office of Scientific Research (AFOSR) under Grants FA9550-08-1-0351 and FA9550-09-1-0035; a gift from Cisco Systems, Inc.; and the A. van Dam Fund. This work was supported in part by grants from the National Science Foundation (NSF) under Grants CNS-0964683 and CNS-1053512; and by grants from the U.S. Air Force Office of Scientific Research (AFOSR) under Grants FA9550-08-1-0351 and FA9550-09-1-0035; a gift from Cisco Systems, Inc.; and the A. van Dam Fund.

Manuscript received May 23, 2011; accepted January 21, 2012; approved by the Communications Policy Committee. This work was supported in part by the National Science Foundation (NSF) under Grants CCF-0826024, CNS-0835949, CNS-0964683, and CNS-1053512; by the U.S. Air Force Office of Scientific Research (AFOSR) under Grants FA9550-08-1-0351 and FA9550-09-1-0035; a gift from Cisco Systems, Inc.; and the A. van Dam Fund. This work was supported in part by grants from the National Science Foundation (NSF) under Grants CNS-0964683 and CNS-1053512; and by grants from the U.S. Air Force Office of Scientific Research (AFOSR) under Grants FA9550-08-1-0351 and FA9550-09-1-0035; a gift from Cisco Systems, Inc.; and the A. van Dam Fund.

This work was supported in part by grants from the National Science Foundation (NSF) under Grants CNS-0964683 and CNS-1053512; and by grants from the U.S. Air Force Office of Scientific Research (AFOSR) under Grants FA9550-08-1-0351 and FA9550-09-1-0035; a gift from Cisco Systems, Inc.; and the A. van Dam Fund.

© 2012 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the copyright owner.

Fig. 1. *FSR* architecture.

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 ·
Katja Obraczka

Published on line: 17 August 2011
© Springer Science+Business Media, LLC 2011

Abstract—We present an efficient message delivery framework, called *McDeHa*, which enables communication in an internet connecting heterogeneous networks that is prone to disconnections in connectivity. *McDeHa* is implemented using the RPL routing protocol. *McDeHa* includes its ability to handle gaps in available destinations. It also employs opportunistic routing to support nodes with episodic connectivity. One of *McDeHa*'s key features is that *McDeHa* can adapt to any changes in the network and can easily be deployed to make two networks inter-operate or to connect to the backbone network. The network is able to store data destined to temporarily unavailable nodes till the time of their expiry. This time period depends upon current storage availability as well as quality-of-service needs (e.g., delivery delay bounds) imposed by the application. We showcase *McDeHa*'s ability to operate in environments consisting of a diverse set of interconnected networks and evaluate its performance through extensive simulations using a variety of metrics with realistic synthetic and real mobility traces. Our results show significant decreases in average delivery ratio and a significant decrease in average delivery delay in the face of episodic connectivity. We also demonstrate that *McDeHa* supports different levels of quality-of-service through traffic differentiation and message prioritization.

Keywords—Disruption tolerance · Episodic connectivity · Heterogeneous networks · Node relaying · Store-and-forward · DTN routing

1 Introduction

It is envisioned that the Internet of the future will be highly heterogeneous not only due to the wide variety of end devices it interconnects, but also in terms of the underlying network technologies. Figure 1 shows the networks that range from wired and wireless backbones (e.g., commodity wireless mesh networks) to wireless infrastructure-based and ad-hoc networks (e.g., MANETs). On the other hand, current and emerging applications, such as emergency response, environmental monitoring, smart environments (e.g., smart offices, homes, sensors, etc.), and vehicular networks, require a reliable, fault-tolerant, and rapidly long-lived disruptions in connectivity. The resulting disruption- or delay-tolerant networks (DTNs) will likely become an important component of future internetworks.

Seamless interoperability among heterogeneous networks is a challenging problem as these networks may have very different characteristics. Node diversity may also

R. N. B. Rais (✉)
COMSATS Institute of Information Technology (CIIT), Lahore, Pakistan
e-mail: naveedrais@ciitlhr.ciit.edu.pk

T. Turletti
INRIA, Sophia Antipolis, France
e-mail: thierry.turletti@sophia.inria.fr

K. Obraczka
University of California, Santa Cruz, CA, USA
e-mail: katja@ucsc.edu

Springer

Augmenting Data Center Networks with Multi-Gigabit Wireless Links

Daniel Halperin¹ · Srikanth Kandula¹ · Jitendra Padhye¹ · Paramvir Bahl² · and David Wetherall¹
¹Microsoft Research and University of Washington

Abstract—The 60 GHz wireless technology that is now emerging has the potential to revolutionize data center (DC) networks. By experimenting with protocols and network architectures, we find that 60 GHz links are well-suited to a deployment of 60 GHz links contrary to concerns about interference and link reliability. Using directional antennas, many wireless links can be established between a top-of-the-rack (ToR) switch and a server rack (SpR) switch. The wired DC network can be used to sidebar several common wireless problems. By analyzing performance traces of DC traffic for four real applications, we show that adding a small amount of 60 GHz wireless links to a DC network with a well-designed wired DC network can improve performance. However, to be of significant value, we find that one hop indirect routing is needed. Interestingly, the 60 GHz links are not always the best choice. We present a design that uses DC traffic levels to select and add flyways to the wired DC network. Trace-driven evaluations show that network performance can be improved by up to 45% in 95% of the cases, with just one wireless device per ToR switch. With two devices, in 40% of the cases, the performance is identical to that of a non-overhauled network.

Categories and Subject Descriptors—C.2.1 [Computer Communication Networks]: Network Architecture and Design—Wireless Communication

General Terms

Design, Experimentation, Measurement, Performance

1. INTRODUCTION

Millimeter wavelength wireless technology is rapidly being developed. Spectrum between 57–64 GHz, colloquially known as the 60 GHz band, is available with wide bandwidth for 802.11ay at 2.4 GHz, and supports devices with multi-Gbps data rates. Furthermore, 60 GHz devices with directional antennas can be deployed densely, because they can be placed close together without frequency interference. The VLSI technology has now matured to the point where 60 GHz radio hardware can be built using CMOS technology, and components can be manufactured for less than \$100 per unit and less than \$10 per unit at OEM quantities. In summary, 60 GHz technology can lead to dense, high-quality wireless connectivity at low costs.

Permitting to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided the copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright © 2011 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the copyright owner.

Transferring the DC network to a 60 GHz wireless network has been studied to keep costs down [13]. For example, a typical DC rack comprises 40 machines connected to a top-of-the-rack (ToR) switch via 60 GHz links. The ToR switch connects to an aggregation switch (60 GHz links). The aggregation switch then connects to a core switch (no 60 GHz links) with 100 Gbps bandwidth. This link from the ToR to the aggregation switch can be overburdened with a ratio of 1:4. However, each overburdened link is a potential bottleneck. One way to mitigate this is to increase the number of links to lessen this problem by combining many more links and switches with variants of multipath routing so that the core of the network is less congested [14]. However, this approach is very expensive due to large material cost and implementation complexity [15]. Some designs require so many wires that cabling becomes a challenge [11], and others require optical [9] approaches.

In prior work [13], we have found for a more modest addition of links to relieve hotspots and boost application performance. The links, called *flyways*, add extra capacity to the base network (the wired DC network). Flyways are not always needed (e.g., if few ToR switches are hot), a small number of flyways can significantly improve performance, without the cost of building a fully meshed network.

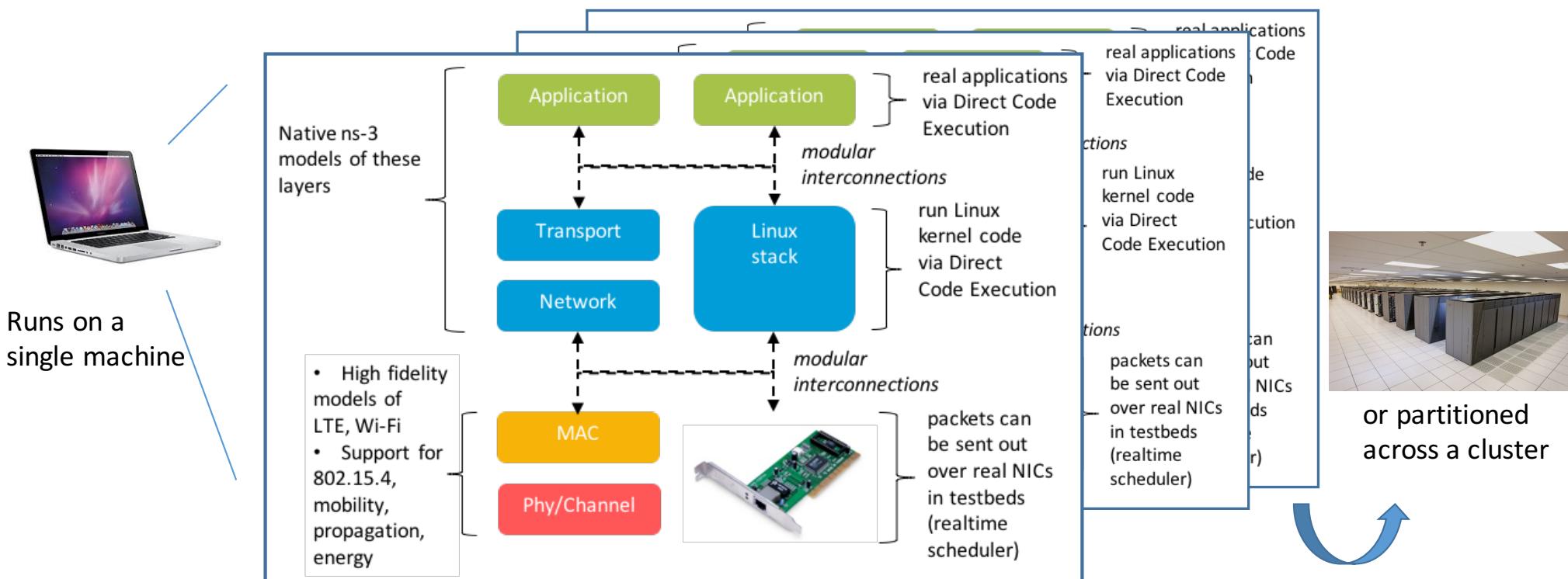
The basic design of a DC network with 60 GHz flyways is as follows. The *wired network* is provisioned for an average traffic load. The *60 GHz network* is provisioned for a peak load. It is equipped with one or more 60 GHz wireless devices, with electronically steerable directional antennas. A central controller monitors traffic patterns, and switches the beams of the wireless devices to wireless links between ToR switches that provide added bandwidth as needed.

Millimeter wave [27], [30] or existing flyways [13] are not the only options. 60 GHz flyways are an attractive choice because wireless devices simply DC upgrade, no wiring changes are needed. Furthermore, 60 GHz flyways can be deployed quickly and inexpensively, unlike optical fiber. Optical fibers are not widely deployed in data centers by consumer applications, while optical switches are not. Wireless devices can introduce additional issues as well—for example, with dynamic topology, the network management may become more

Copyright © 2011 ACM 978-1-4503-0974-0/10/08-10.00.

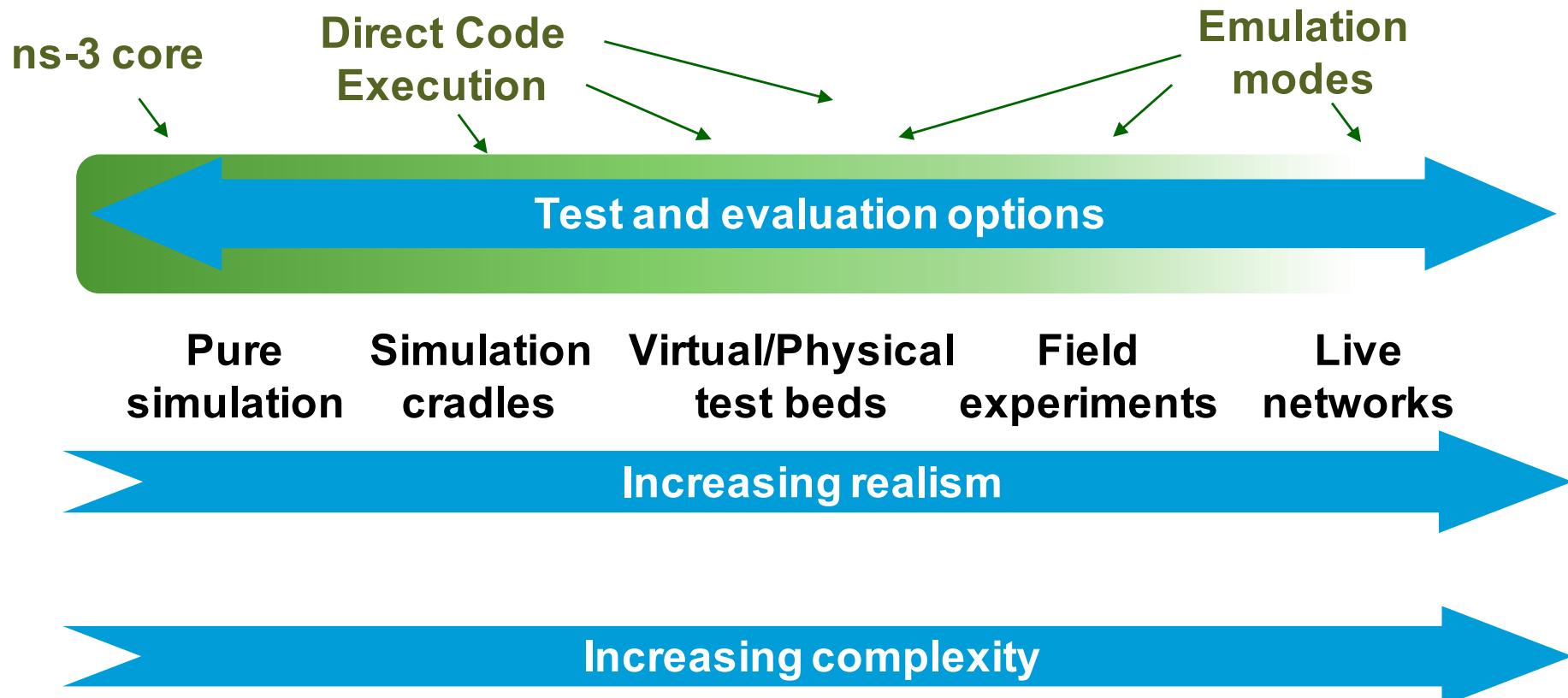
ns-3 overview

- ns-3 is a leading open source, **packet-level network simulator** oriented towards network research, featuring a **high-performance core** enabling **parallelization across a cluster** (for large scenarios), **ability to run real code**, and **interaction with testbeds**



Network performance evaluation options

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



-
- The open-source project

ns-3 main website

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

The screenshot shows the official ns-3 Network Simulator website. At the top, there's a dark header bar with the ns-3 logo on the left and five navigation links: OVERVIEW, RELEASES, DOCUMENTATION, DEVELOPERS, and SUPPORT. Below the header, there's a large green graphic of four 3D bars of increasing height. To the left of this graphic, a text box explains what ns-3 is: "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." To the right of the graphic, a sidebar titled "Recent Posts" lists several news items with dates and brief descriptions.

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...

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](#)

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
- Summer projects (Google Summer of Code, ESA Summer of Code in Space, others...)

Maintainers, Authors, Users

- ~10-15 maintainers at any given time
- 191 authors credited in AUTHORS file
- Over 6000 subscribers to ns-3-users Google Groups forum
- Over 1500 subscribers to ns-developers mailing list
- Various project forks exist (on Github and elsewhere)

Contributed code and associated projects

The screenshot shows the "Overall ndnSIM documentation" page. At the top, there's a navigation bar with links for "ndnSIM" and "ndnSIM API". Below the navigation, there's a search bar with placeholder text "Enter search terms or a module, class or function name." and a "Go" button. The main content area has a heading "Welcome to ndnSIM NS-3 based NDN simulator" and a paragraph inviting users to join the mailing list. It also contains a "Contents" sidebar with sections like "Introduction", "Getting Started", and "ndnSIM Helper".

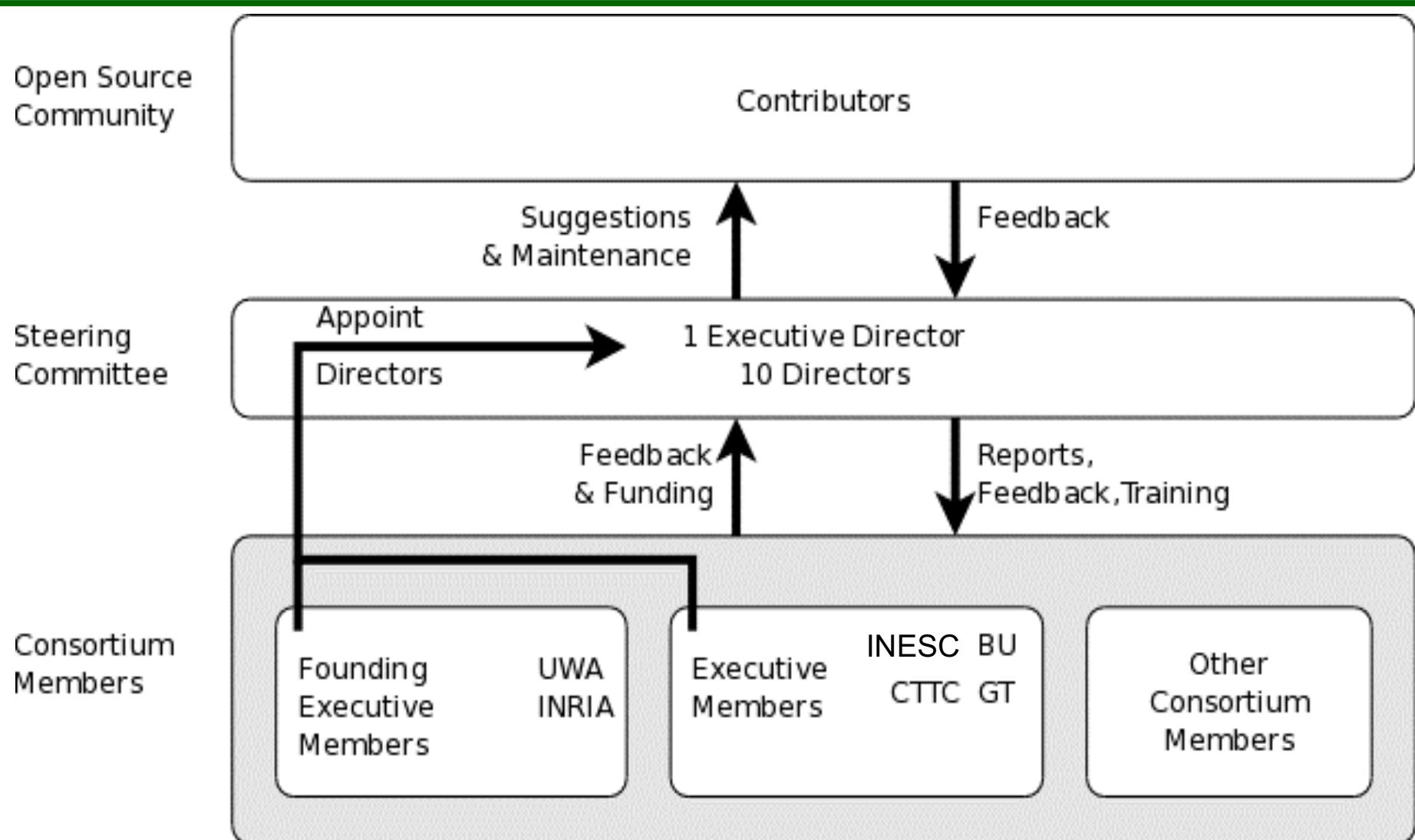
The screenshot shows the "mptcp-ns3" project page. At the top, there's a logo and the text "implement multipath TCP on ns-3". Below the logo, there's a navigation bar with tabs for "Project Home", "Downloads", "Wiki", "Issues", and "Source". The "Project Home" tab is selected. The main content area has a "Project description" section with a brief overview of the project's focus on developing Multipath TCP for ns-3. It also includes sections for "Project Information", "Code license", "Labels", "Members", and "Links".

The screenshot shows the "Decentralized Systems and Network Services Research Group - TM & SC" page. At the top, there's a KIT logo and a navigation bar with links for "HOME", "DEUTSCH", "LEGAL", "SITEMAP", and "KIT". The main content area has a "Overview" section with information about the group's work on "PhySimWiFi for NS-3". It includes contact details for Jens Mittag and Stylianos Papanastasiou, and a "Publications" section listing various research papers and projects.

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.

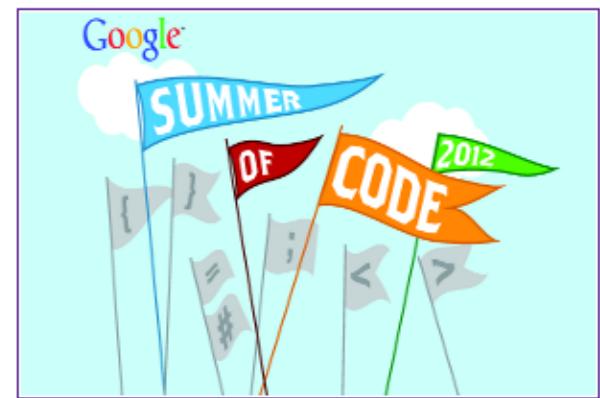
ns-3 Consortium governance



Acknowledgment of support



**Georgia
Tech**



ns-3 Training, June 2017

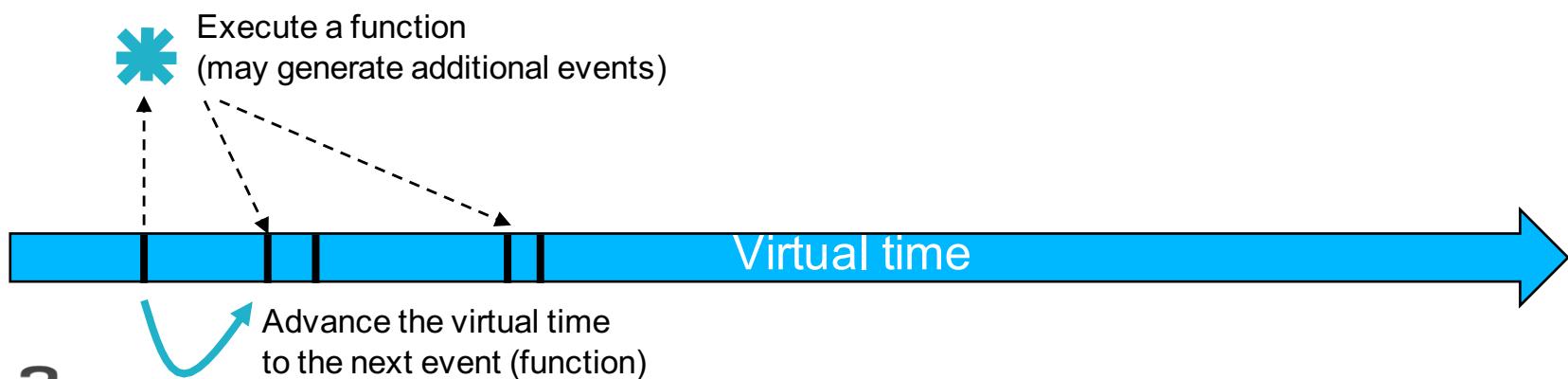
-
- Software overview

Software overview

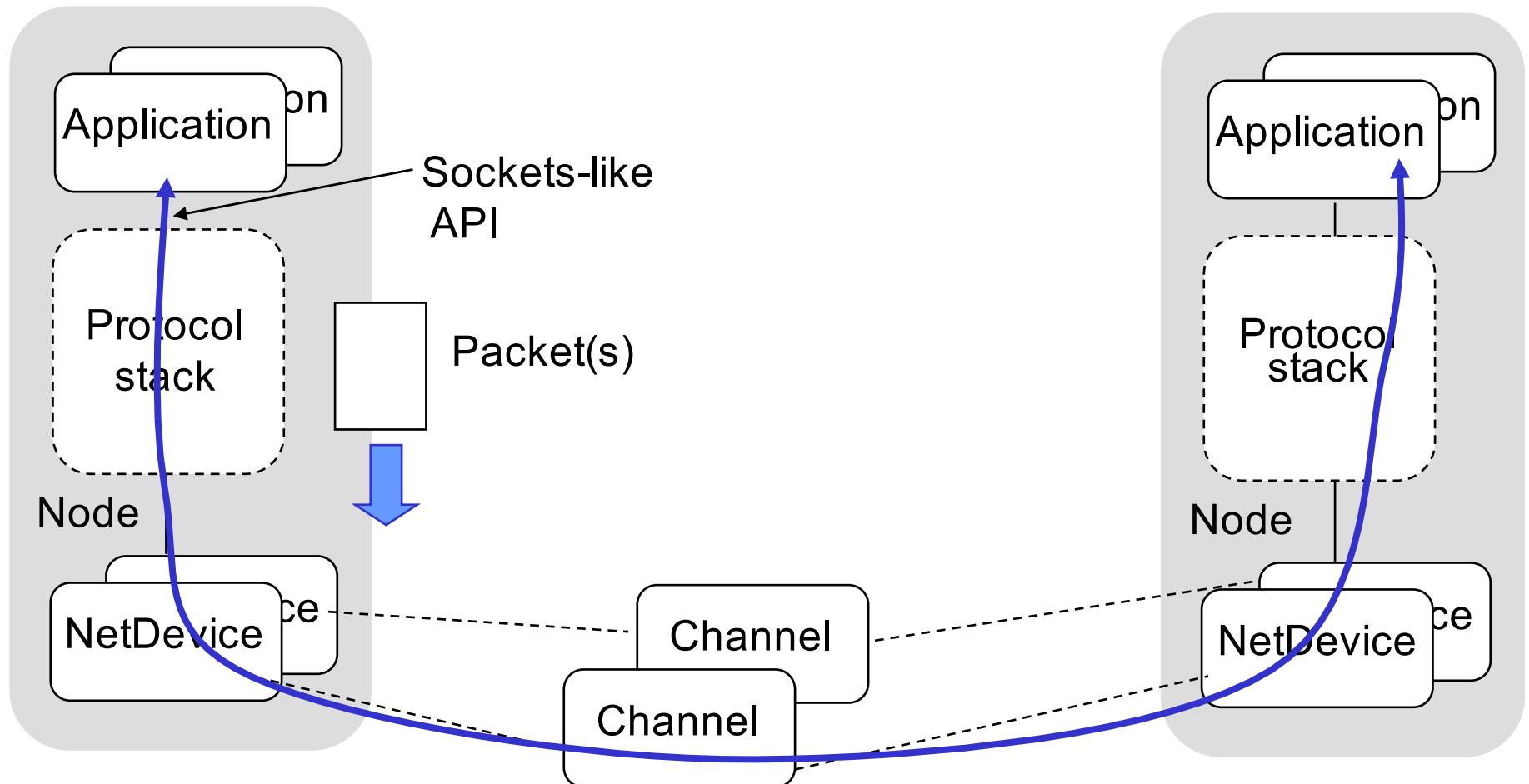
- ns-3 is written in C++, with bindings available for Python
 - simulation programs are C++ executables or Python programs
 - ~350,000 lines of C++ (cloc estimate)
 - almost exclusively C++98, beginning to use C++11
- ns-3 is a GNU GPLv2-licensed project
- ns-3 is mainly supported for Linux, OS X, and 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

ns-3 does not have a graphical IDE

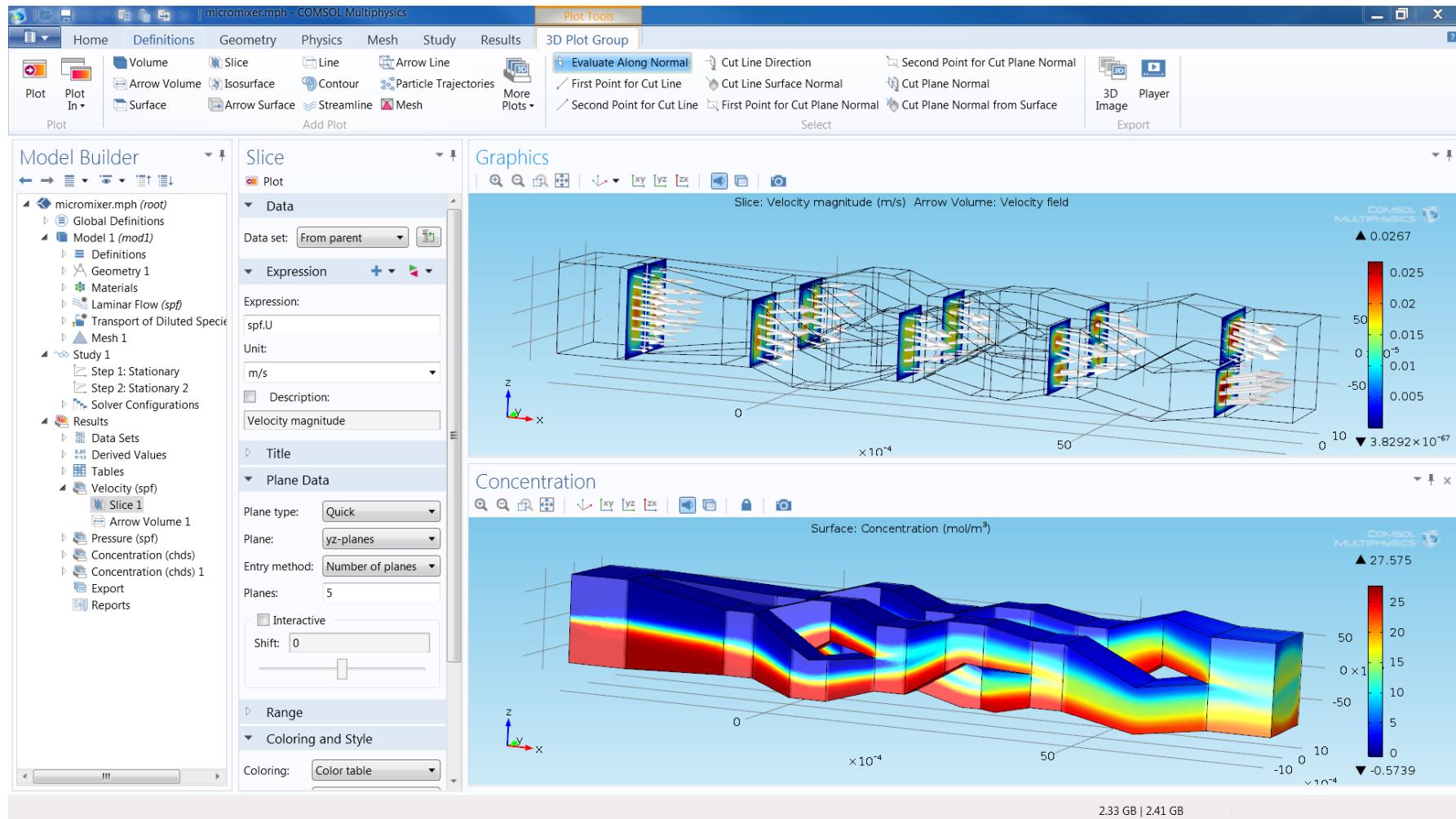


Figure source: <https://www.comsol.com/comsol-multiphysics>
ns-3 Training, June 2017

ns-3 not written in a high-level language

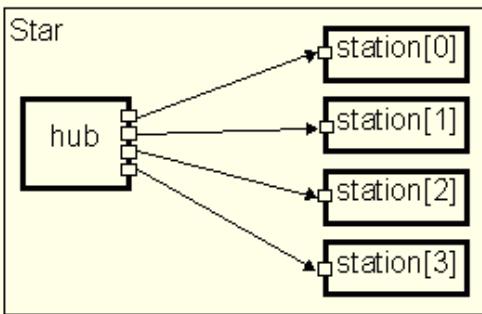
Submodule vectors, gate vectors and multiple connections are illustrated in the following example:

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

simple Station //...

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

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



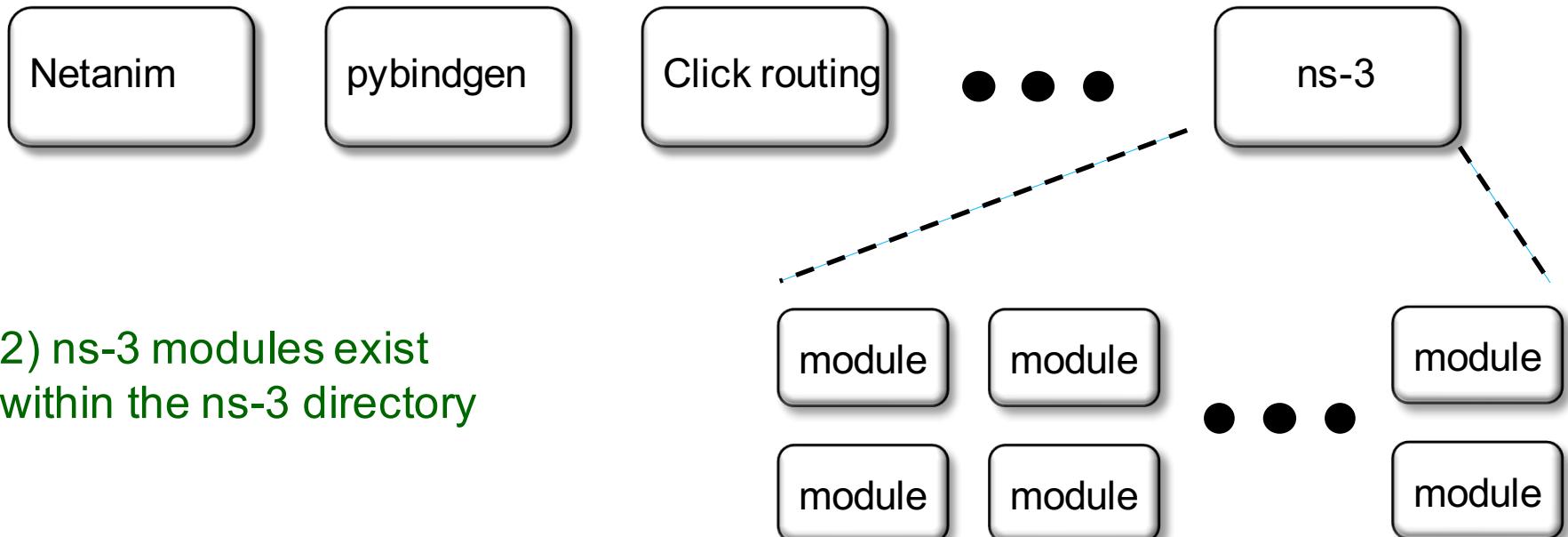
Example of OMNeT++ Network Description (NED) language

Figure excerpted from <http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm>

Software organization

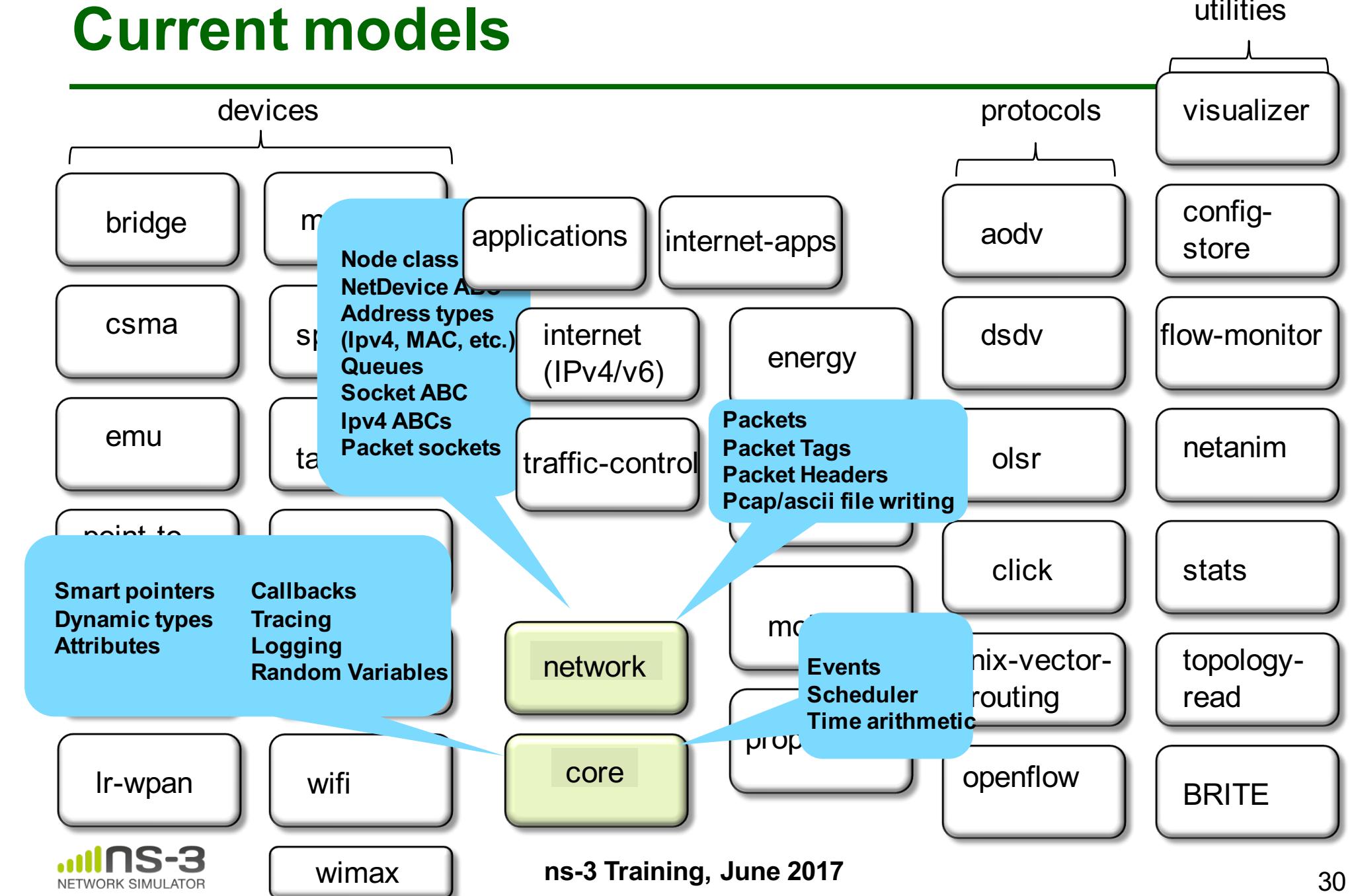
- Two levels of ns-3 software and libraries

- 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



Module organization

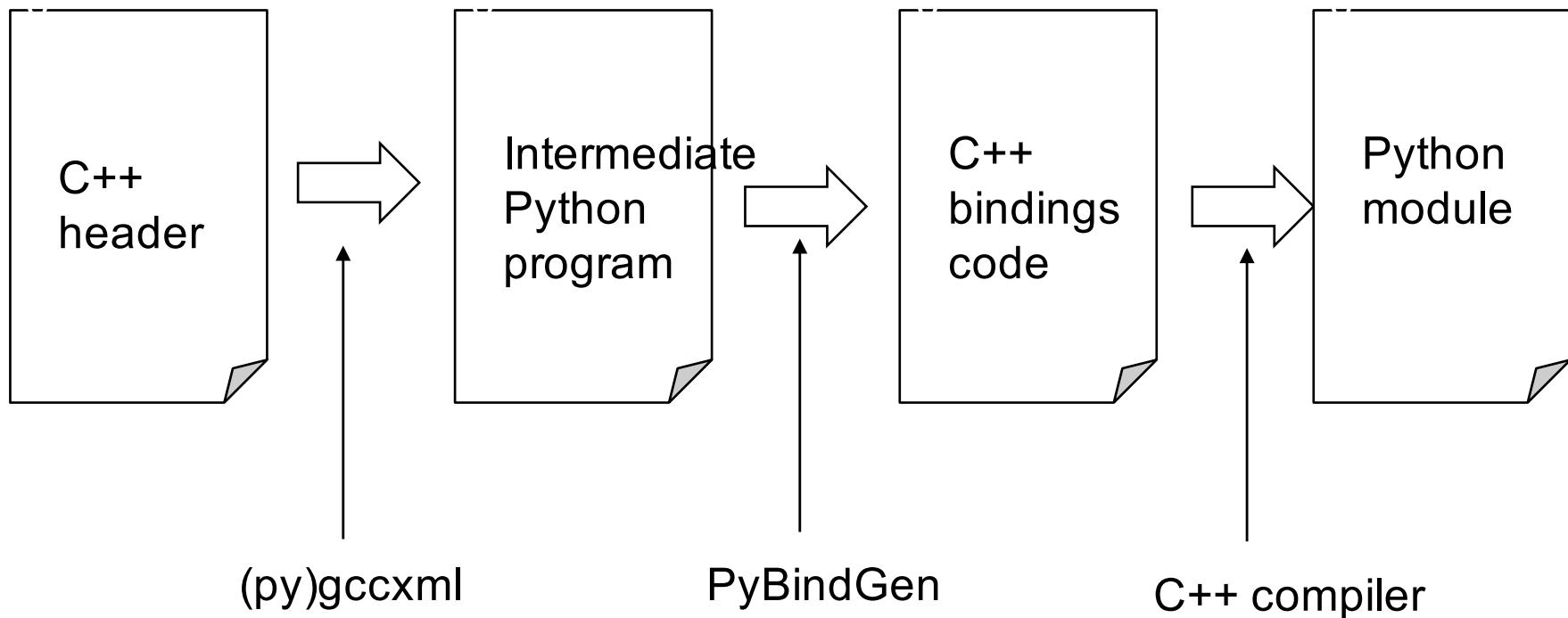
- models/
- examples/
- tests/
- bindings/
- doc/
- wscript

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

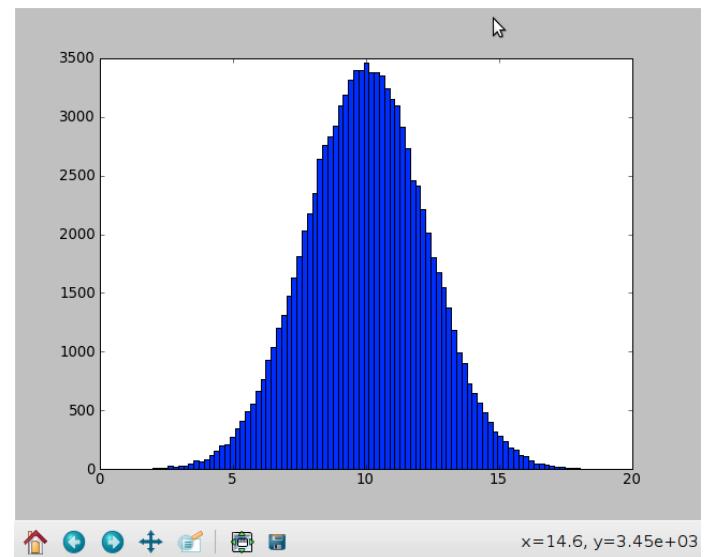
- more sophisticated scenarios and models typically leverage other libraries
- ns-3 main distribution uses optional libraries (libxml2, gsl, mysql) but care is taken to avoid strict build dependencies
- the 'bake' tool (described later) helps to manage library dependencies
- users are free to write their own Makefiles or wscripts to do something special

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()
```

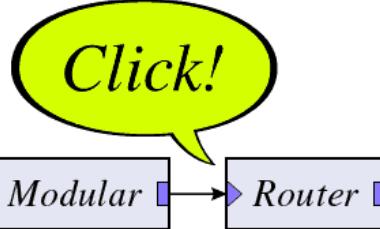


Click Modular Router

Click!

Login

Show pagesource Old revisions Sitemap Recent changes Search



The Click Modular Router Project



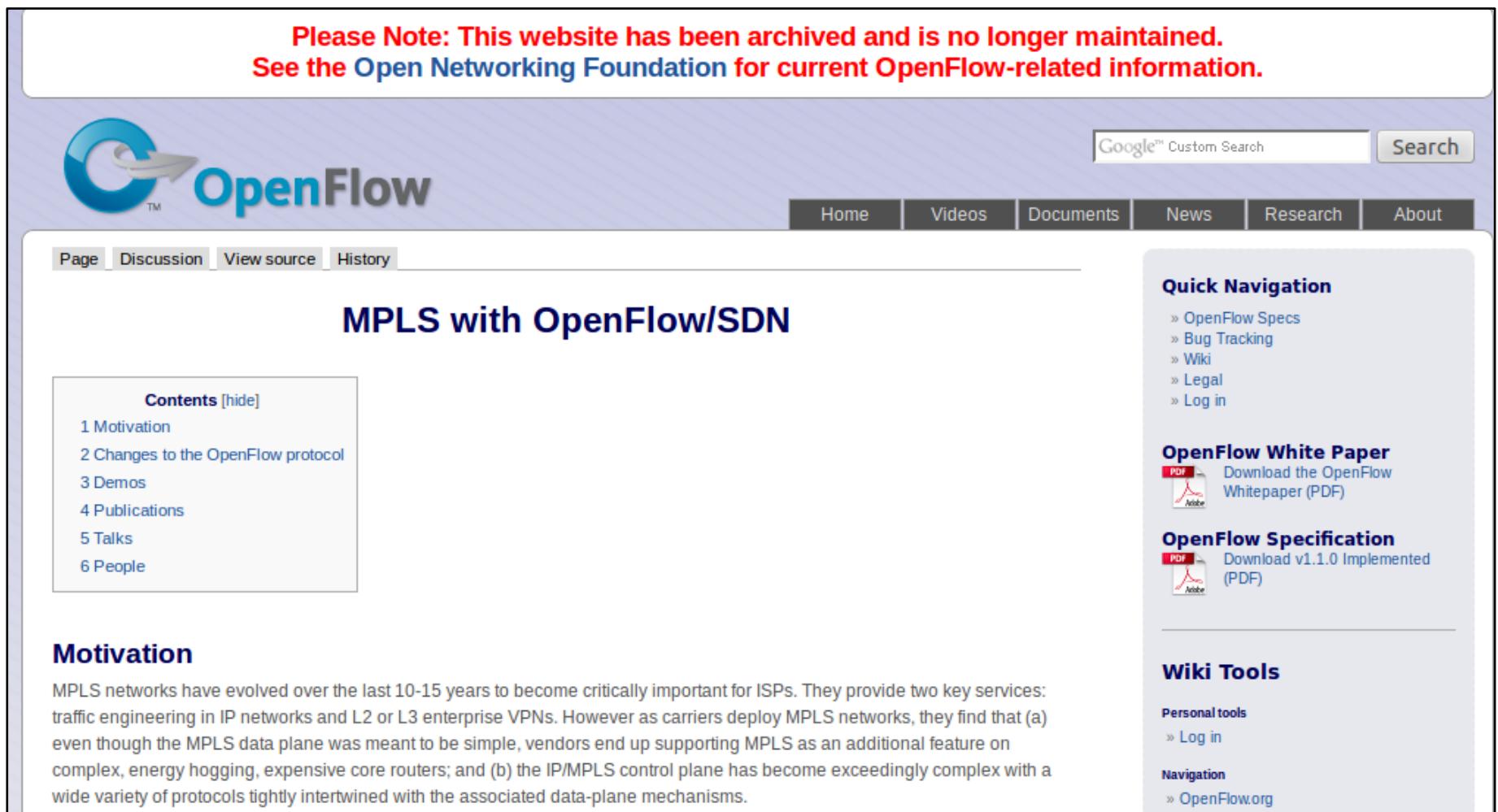
NEWS (September 24, 2011): **Click 2.0.1 released!**

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.



The screenshot shows the homepage of the OpenFlow website. At the top, a red banner displays a note about the website being archived. Below the banner, the OpenFlow logo is visible, followed by a search bar and a navigation menu with links to Home, Videos, Documents, News, Research, and About. A sidebar on the right contains links for Quick Navigation, including OpenFlow Specs, Bug Tracking, Wiki, Legal, and Log in. It also features links for the OpenFlow White Paper (PDF) and the OpenFlow Specification (PDF). The main content area includes a section titled "MPLS with OpenFlow/SDN" and a "Contents" sidebar with links to Motivation, Changes to the OpenFlow protocol, Demos, Publications, Talks, and People.

MPLS with OpenFlow/SDN

Contents [hide]

- 1 Motivation
- 2 Changes to the OpenFlow protocol
- 3 Demos
- 4 Publications
- 5 Talks
- 6 People

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

The screenshot shows the GitHub repository page for `mininet / mininet`. The page includes a navigation bar with links to Explore, Features, Enterprise, Blog, Sign up, and Sign in. Below the navigation is a search bar and a repository summary showing 468 stars and 204 forks. The main content area features a "Link modeling using ns 3" section, a "Contents" sidebar with a table of contents, and a right sidebar with various links and icons.

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

- [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](#)

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

- [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

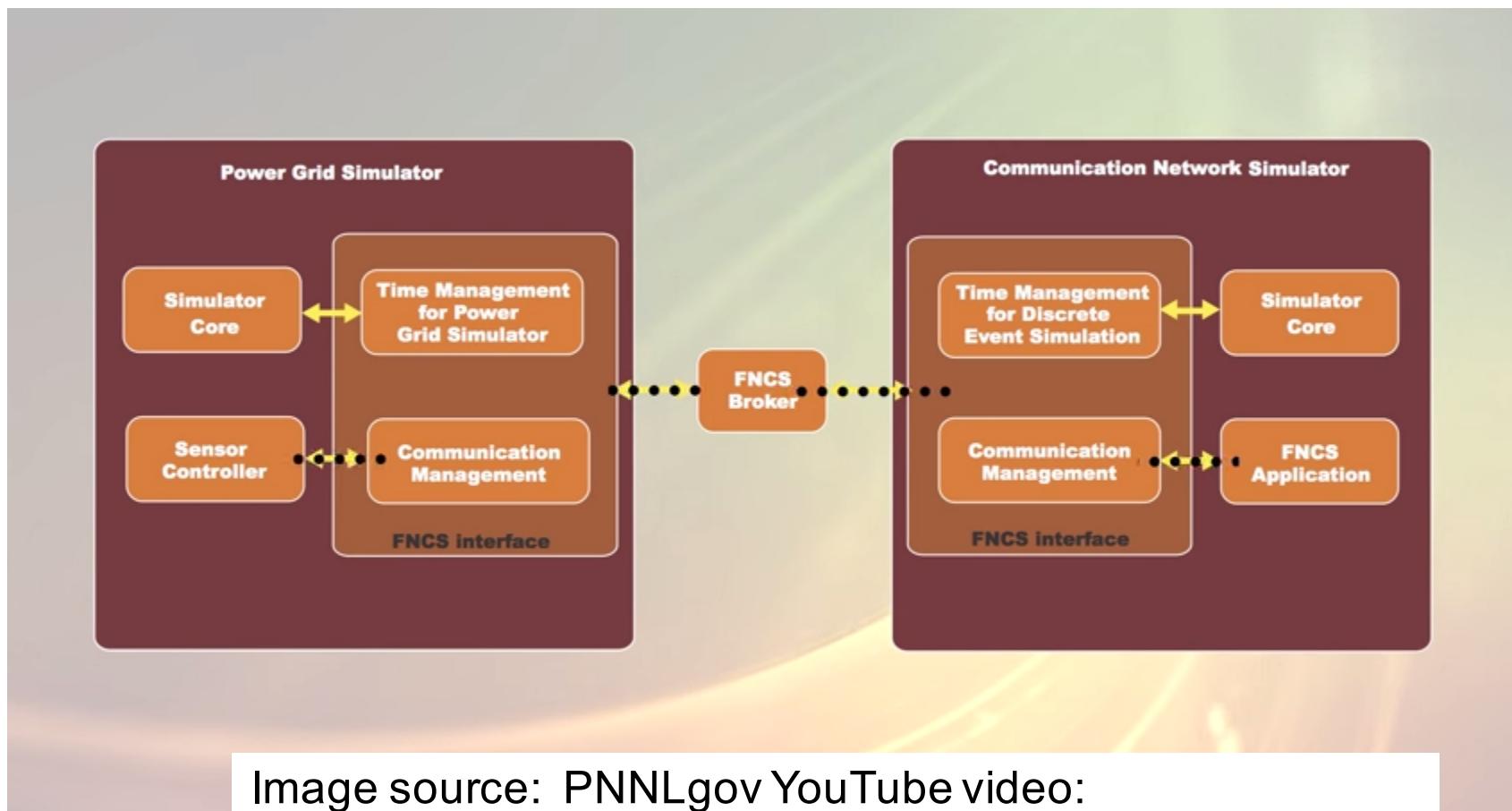


Image source: PNNLgov YouTube video:
Introducing FNCS: Framework for Network Co-Simulation
ns-3 Training, June 2017

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 officially, but some package maintainers exist
- 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