ns-3 Training

ns-3 Annual Meeting

June 2018

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

ns-3 training wiki page

https://www.nsnam.org/wiki/AnnualTraining2018

Log in

Page | Discussion | Read | View source | View history | Search

AnnualTraining2018

Main Page - Current Development - Developer FAQ - Tools - Related Projects - Project Ideas - Summer Projects
Installation - Troubleshooting - User FAQ - HOWTOs - Samples - Models - Education - Contributed Code - Papers

This is the wiki page for the ns-3 training sessions to be held on 11-12 June, 2018 in Surathkal. For a list of all past training sessions, see Training.

Contents

1 Agenda
2 Registration
3 Local Information
4 Additional files
5 Supplementary training videos

Agenda

Training will be held from 09:00-18:00 at NITK Surathkal. Please plan to arrive 15-30 minutes beforehand so we may start at 09:00.

There will be two morning sessions and two afternoon sessions, and canteen lunch provided (lunch break from 12:30-14:00).

See the flyer for more agenda information: https://www.nsnam.org/docs/consortium/training/ns-3-training-2018.pdf

Slides will be posted here once available.

Main page
Recent changes
Random page
Help

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

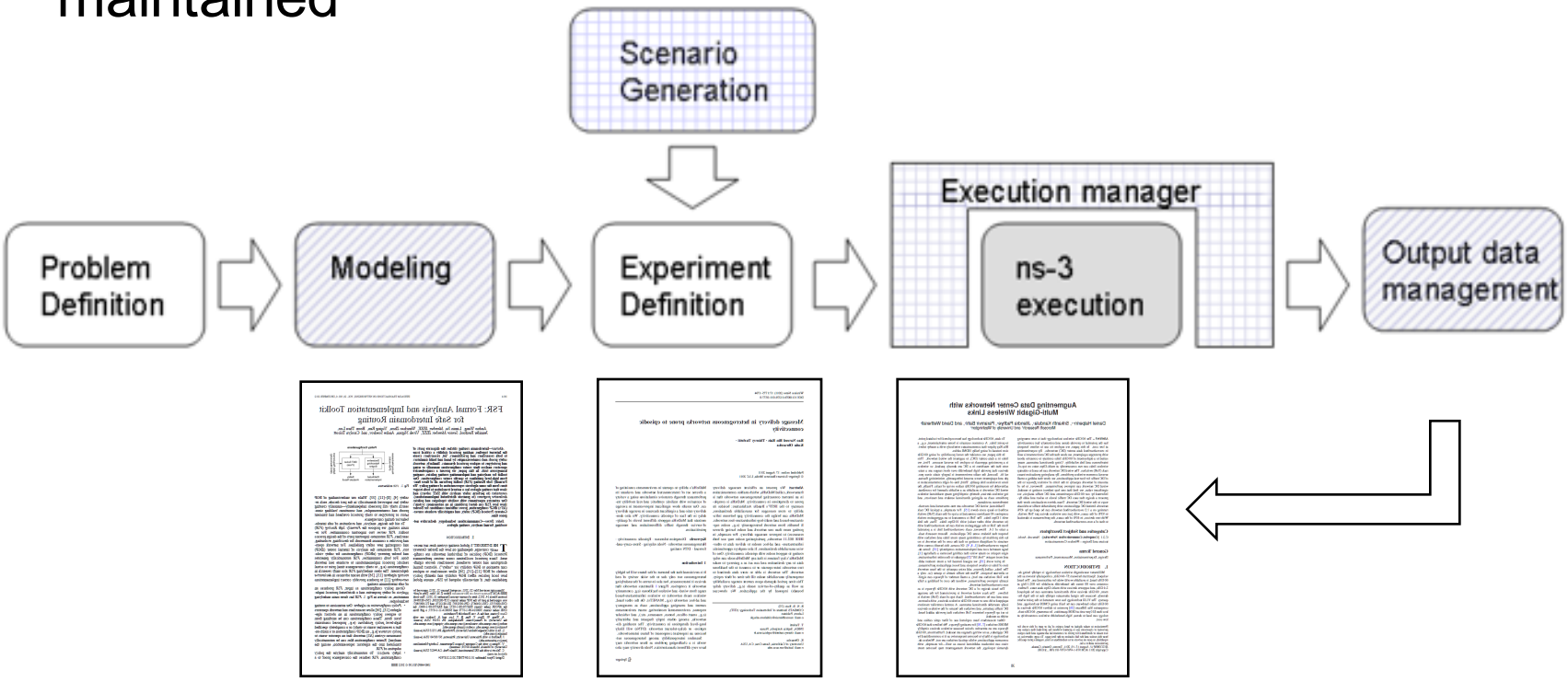
Agenda and Instructors

- Monday: ns-3 overview (T. Henderson)
  - project overview
  - software overview
  - simulator introduction via sample workflows
- Tuesday AM: traffic control (M. Tahiliani)
- Tuesday AM: ns-3 advanced modes: DCE, MPI, emulation (S. Jain and T. Henderson)
- Tuesday PM: Wi-Fi (S. Deronne)
- Tuesday PM: LTE (N. Patriciello)

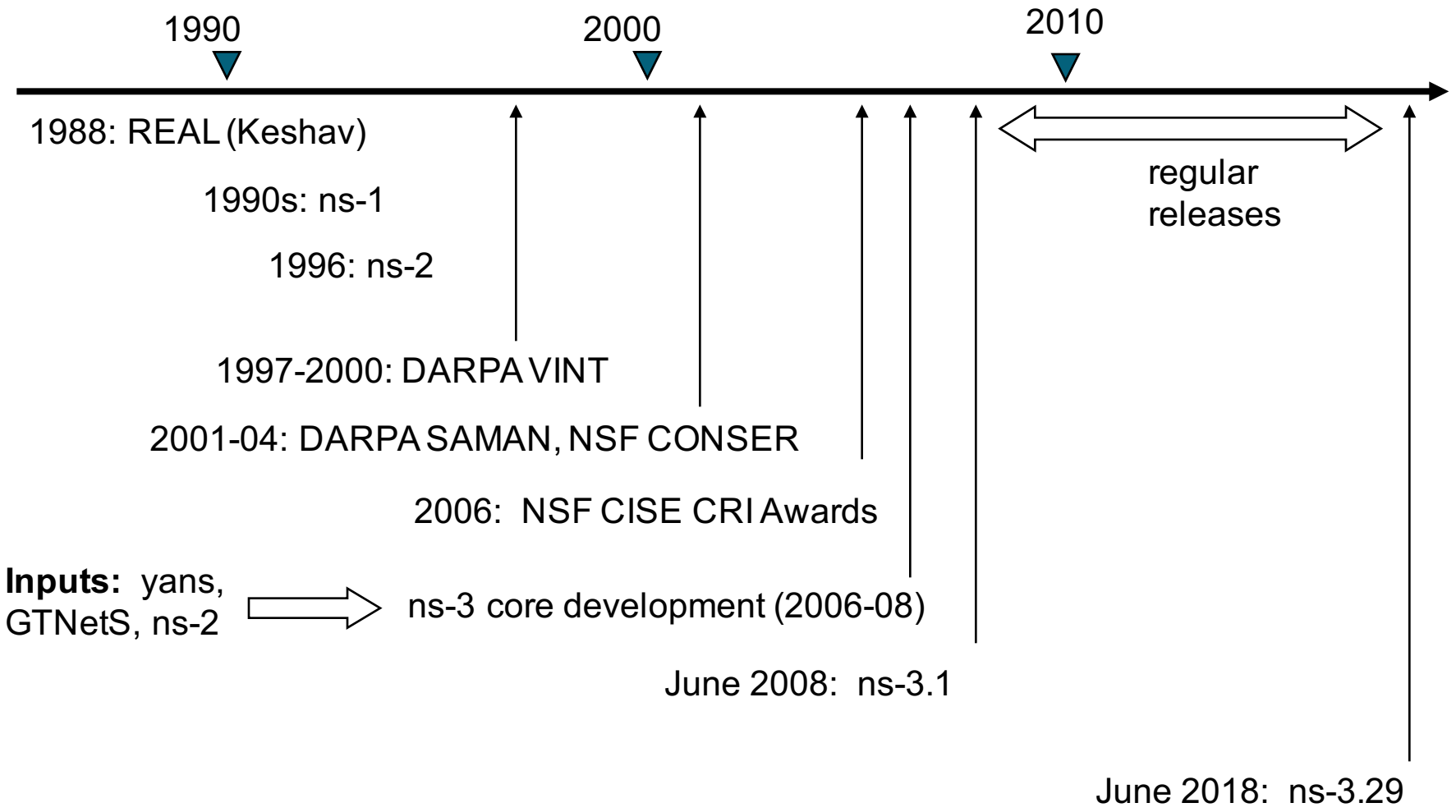
Project overview

ns-3 in a nutshell

- **Purpose:** Tool for network performance analysis
- **Users:** Researchers and academia
- **Development model:** Open source, community maintained

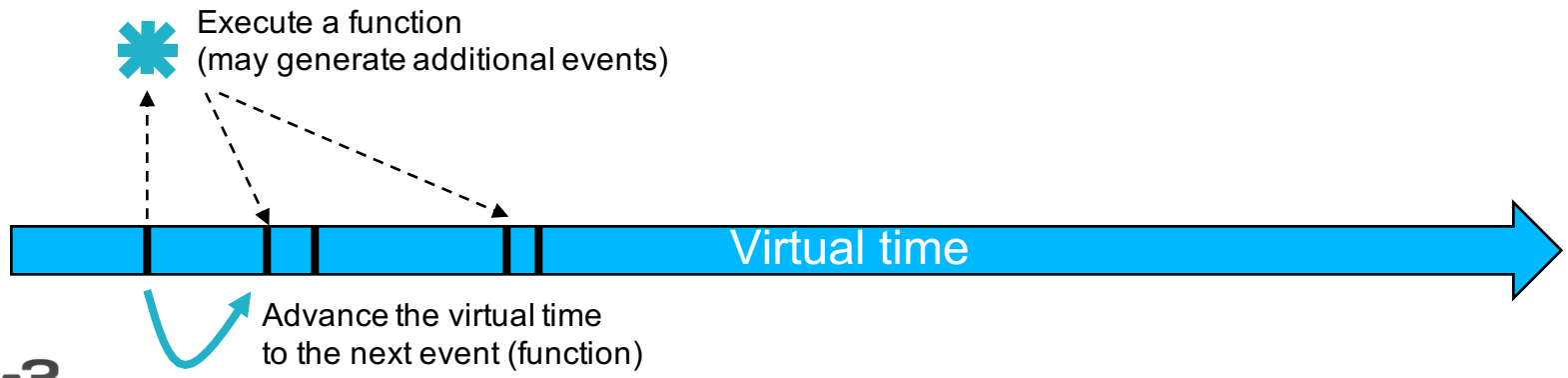


ns timeline

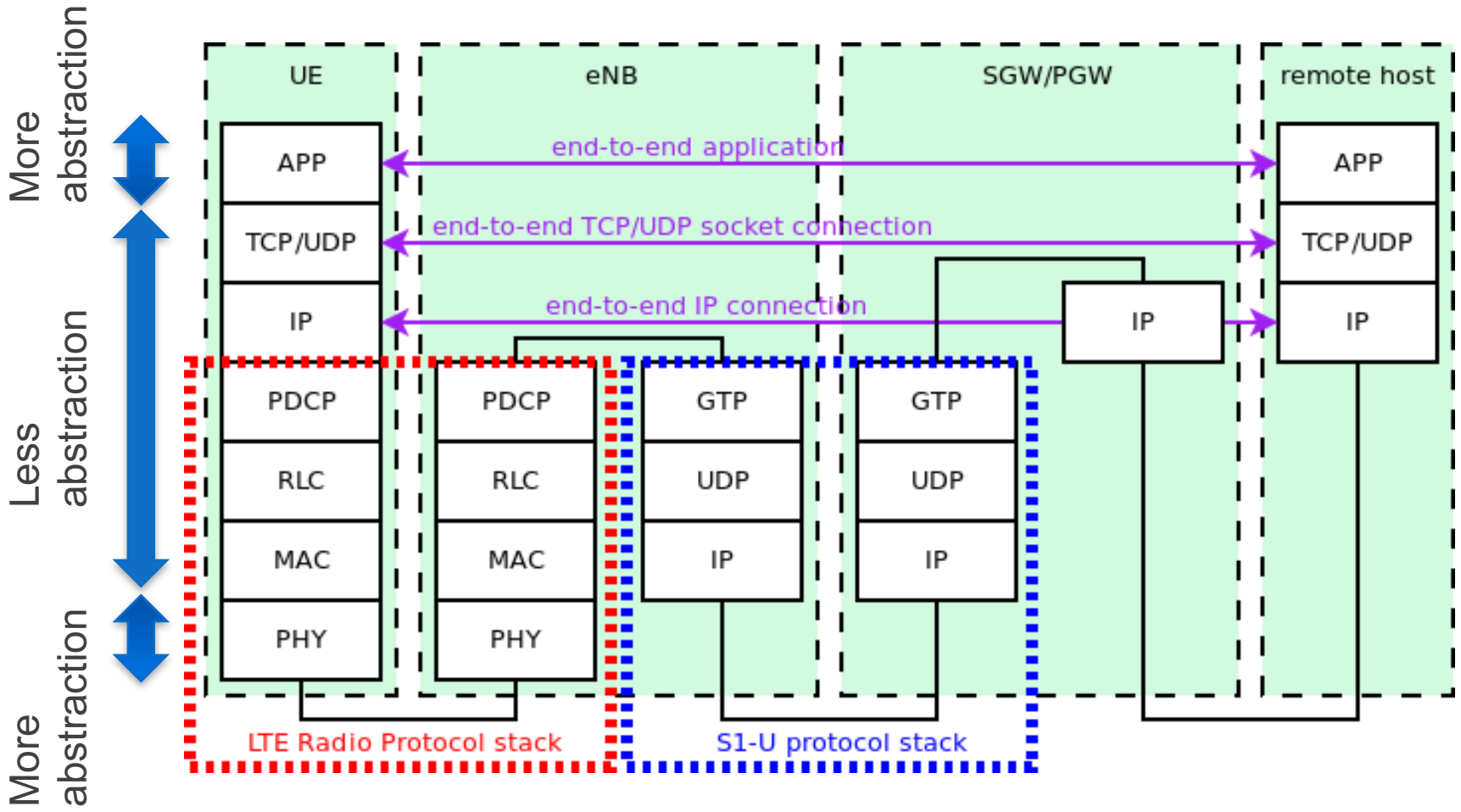

1990
2000
2010
1988: REAL (Keshav)
1990s: ns-1
1996: ns-2
1997-2000: DARPA VINT
2001-04: DARPA SAMAN, NSF CONSER
2006: NSF CISE CRI Awards
regular releases
Inputs: yans, GTNetS, ns-2
ns-3 core development (2006-08)
June 2008: ns-3.1
June 2018: ns-3.29

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() executes a single-threaded event list
- Simulation stops at specific time or when events end

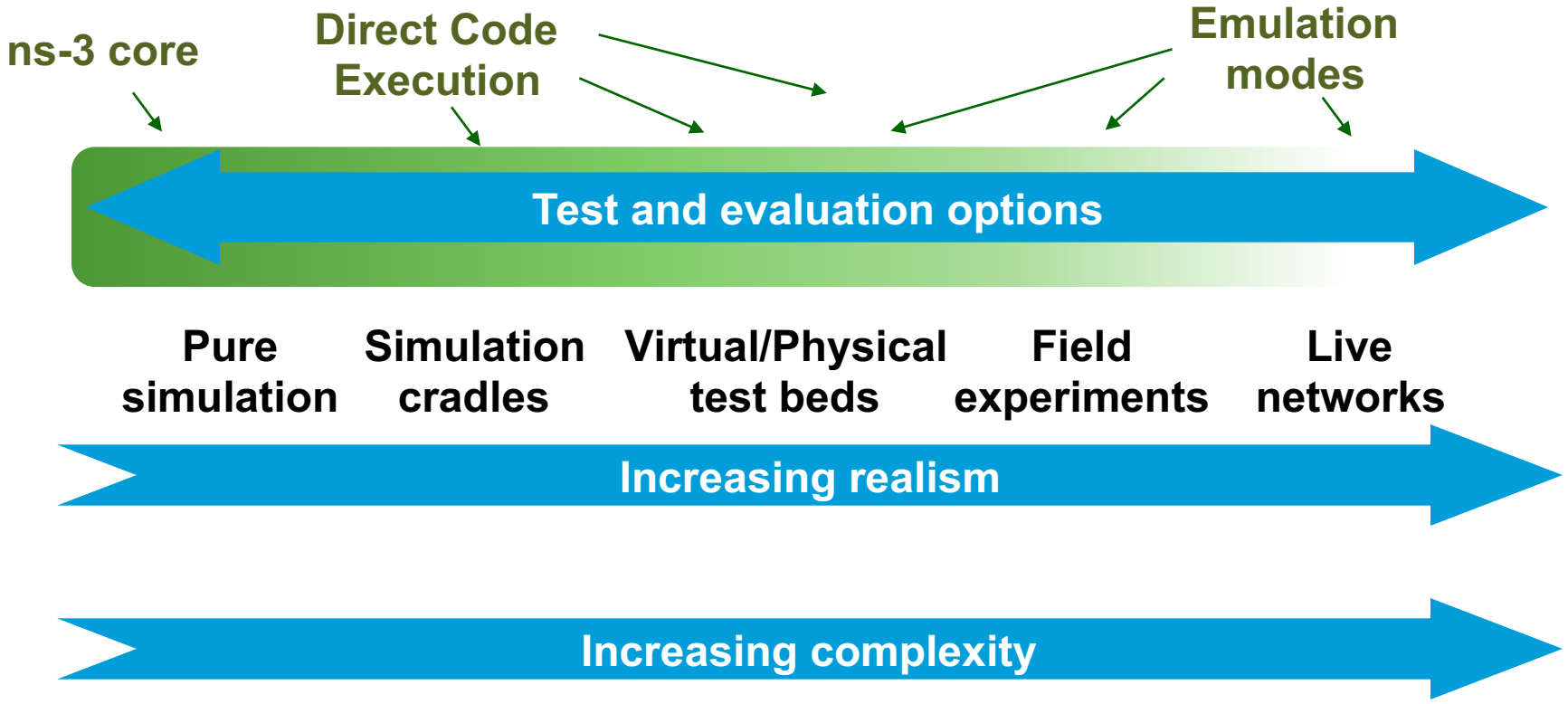


Packet level simulation overview



Network performance evaluation options

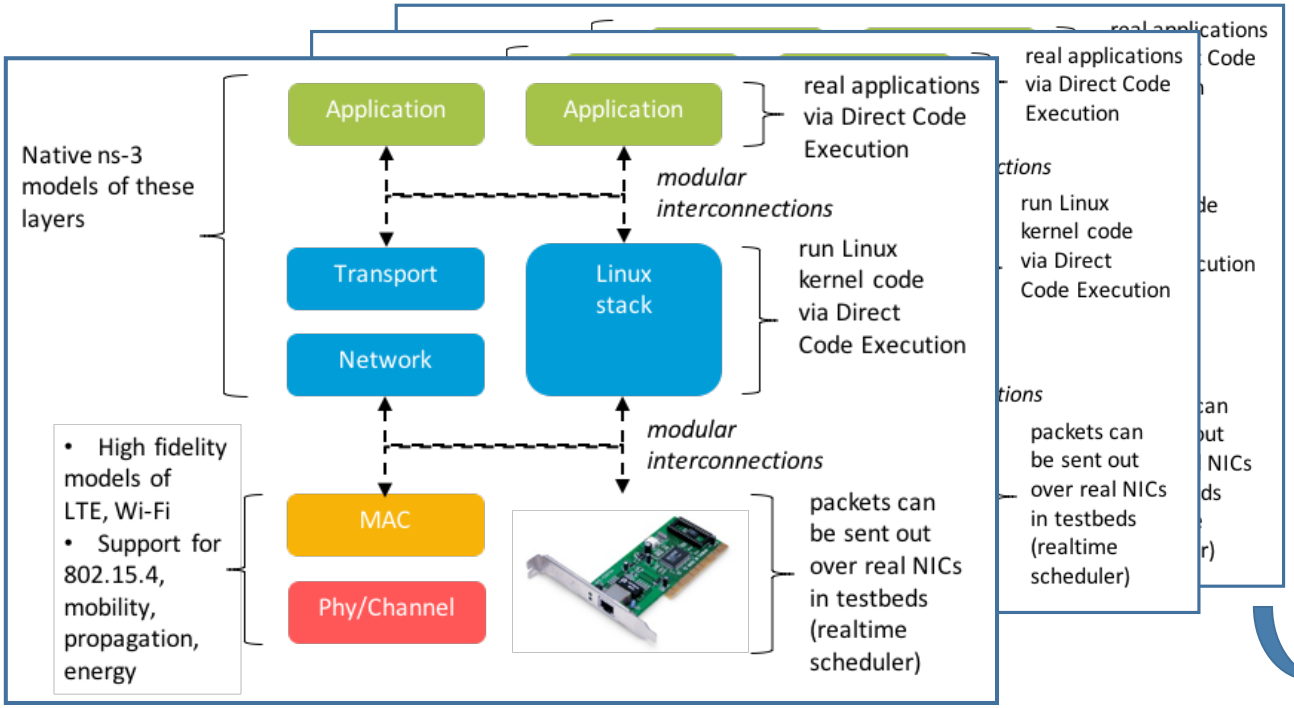
- ns-3 enables researchers to more easily move between simulations, test beds, and experiments



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**

Runs on a single machine



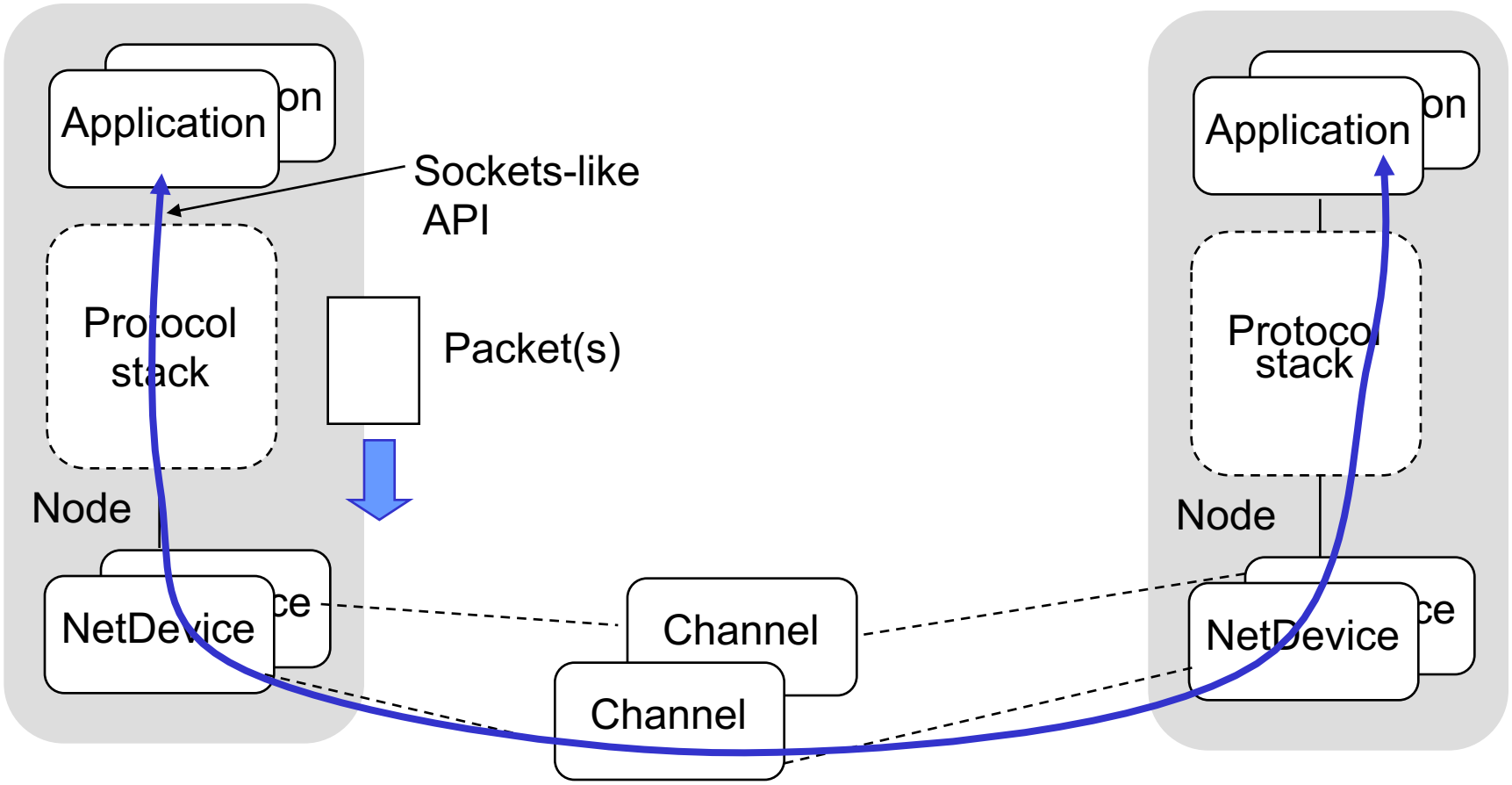
or partitioned across a cluster

- 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 in progress
- ns-3 is not backwards-compatible with ns-2

The basic ns-3 architecture



Software orientation

Key differences from other network simulators:

1) Command-line, Unix orientation
   - vs. Integrated Development Environment (IDE)

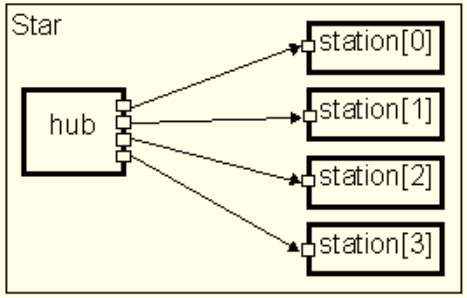
2) Simulations and models written directly in C++ and Python
   - vs. a domain-specific simulation language

ns-3 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

ns-3 does not have a graphical IDE

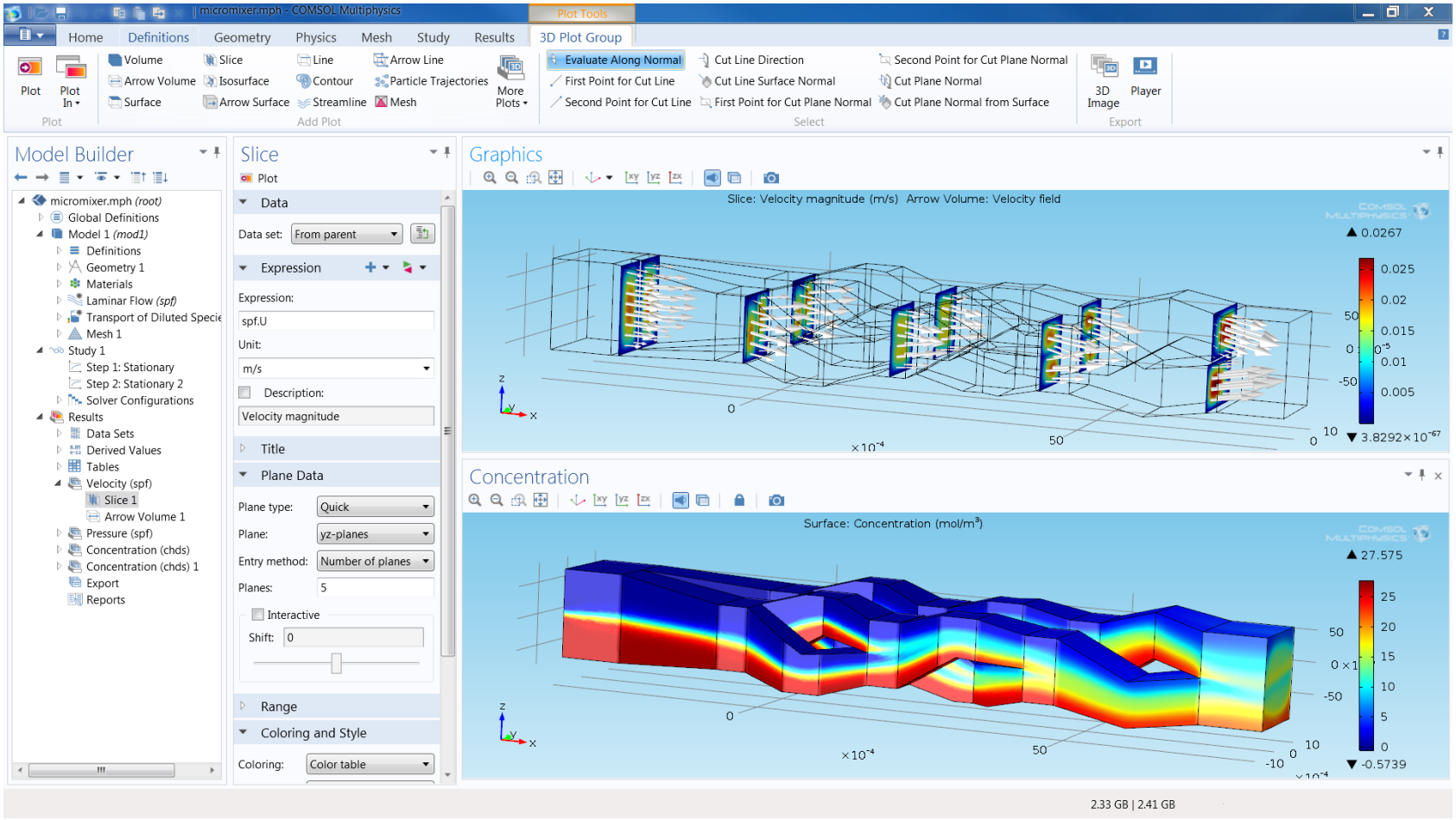
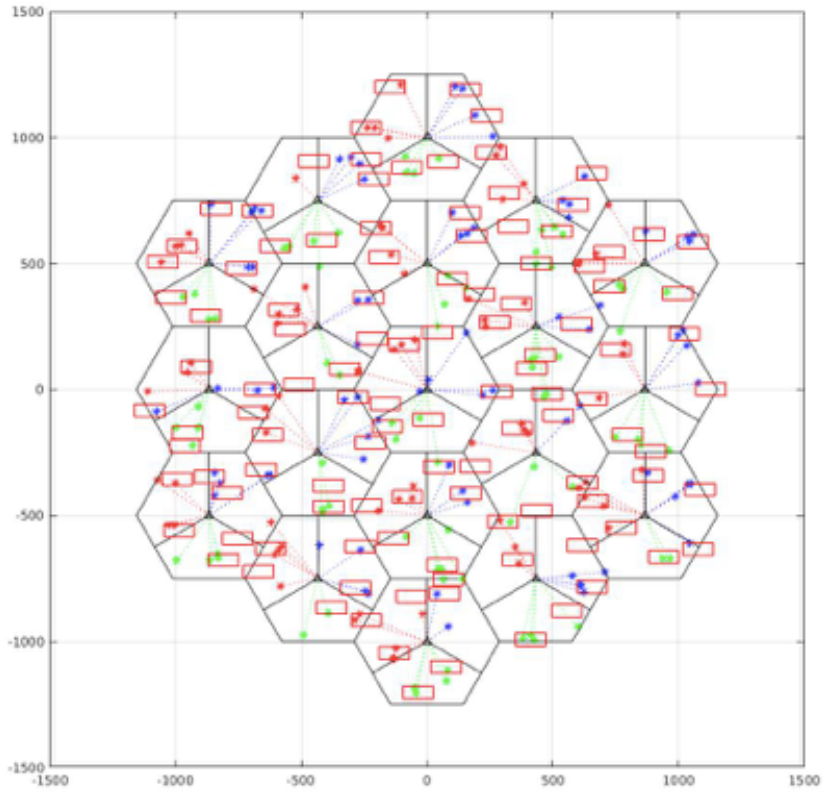
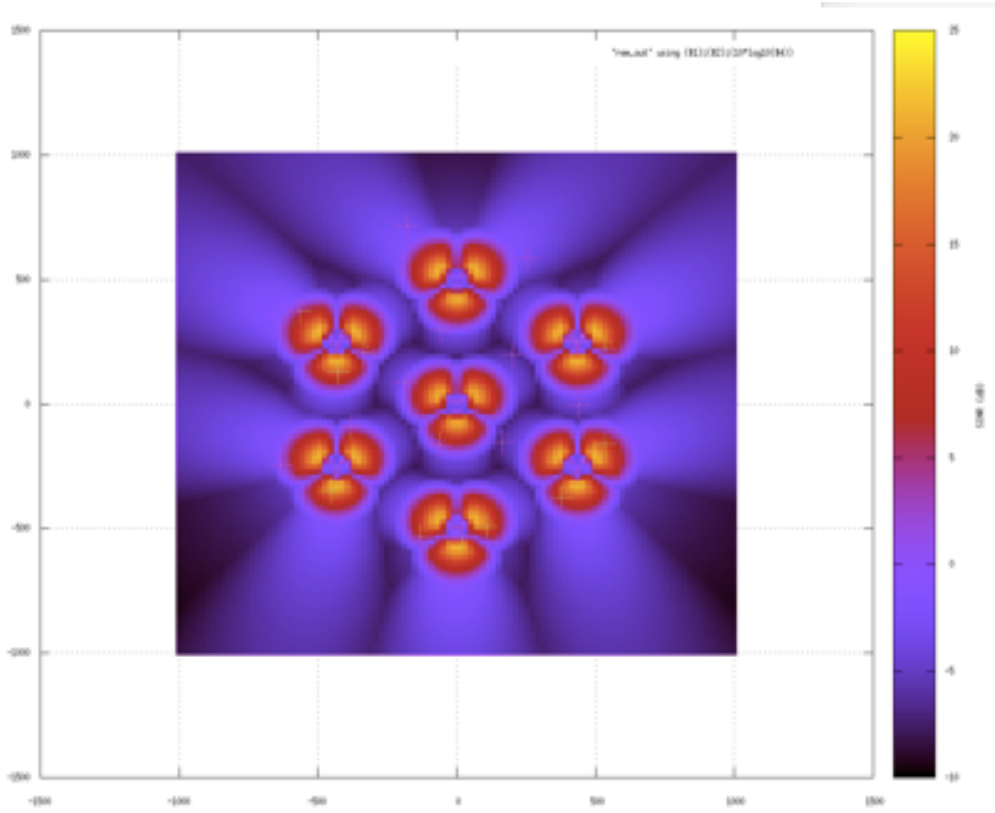


Figure source: https://www.comsol.com/comsol-multiphysics

ns-3 uses outside programs for graphics



Matplotlib

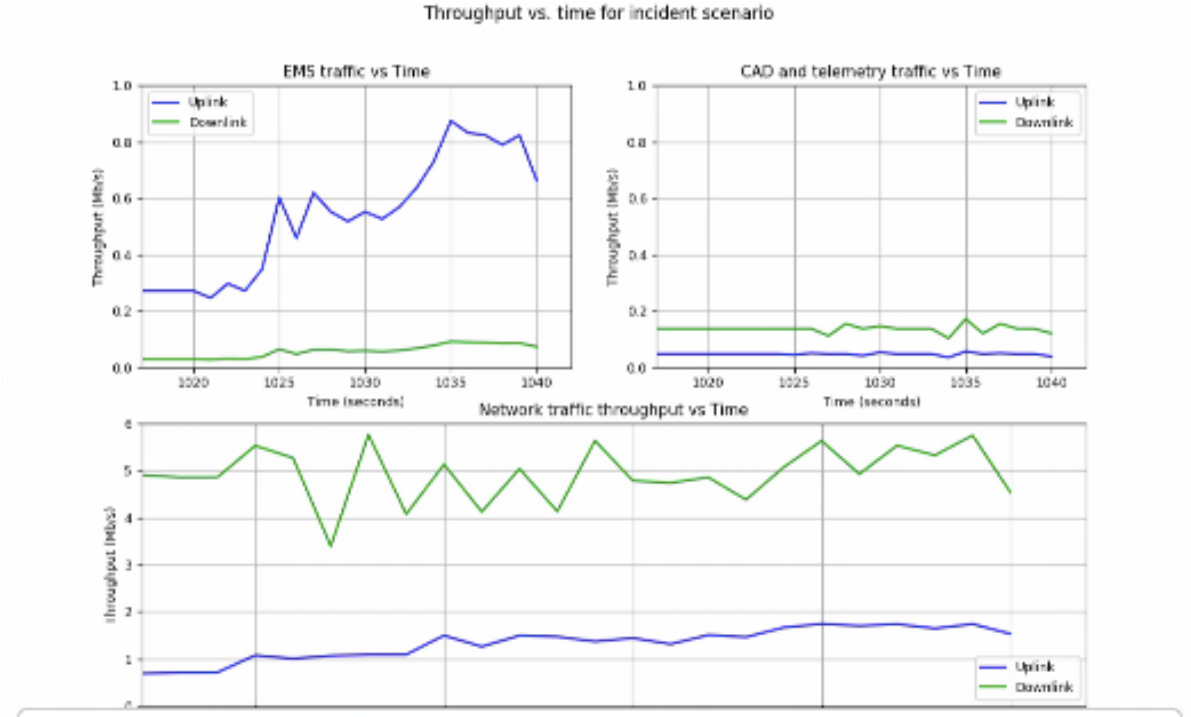


gnuplot

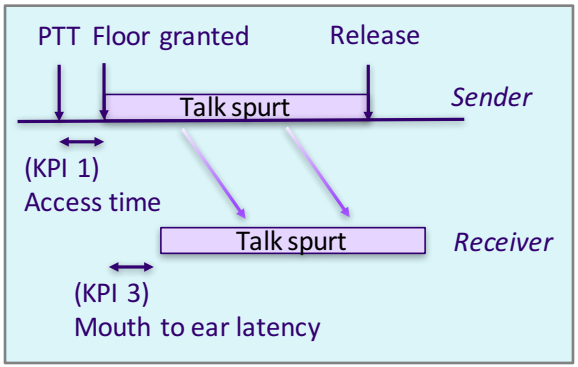
ns-3 uses scripting for plotting

Animated

EmsVideo_1_Server 942.36392953 RX 1012 1061 U 2395
EmsVideo_1_Client 942.37317727 TX 1012 1061 U 2398
EmsVideo_1_Client 942.377 RX 64 113 U 2397
WebBrowsingGraphics_0_Server 942.38092876 TX 1024 1073 U 2399
WebBrowsingGraphics_0_Client 942.394 RX 1024 1073 U 2399
AvlAssetPerimeter_1_Server 942.42492988 RX 1408 1457 U 2401



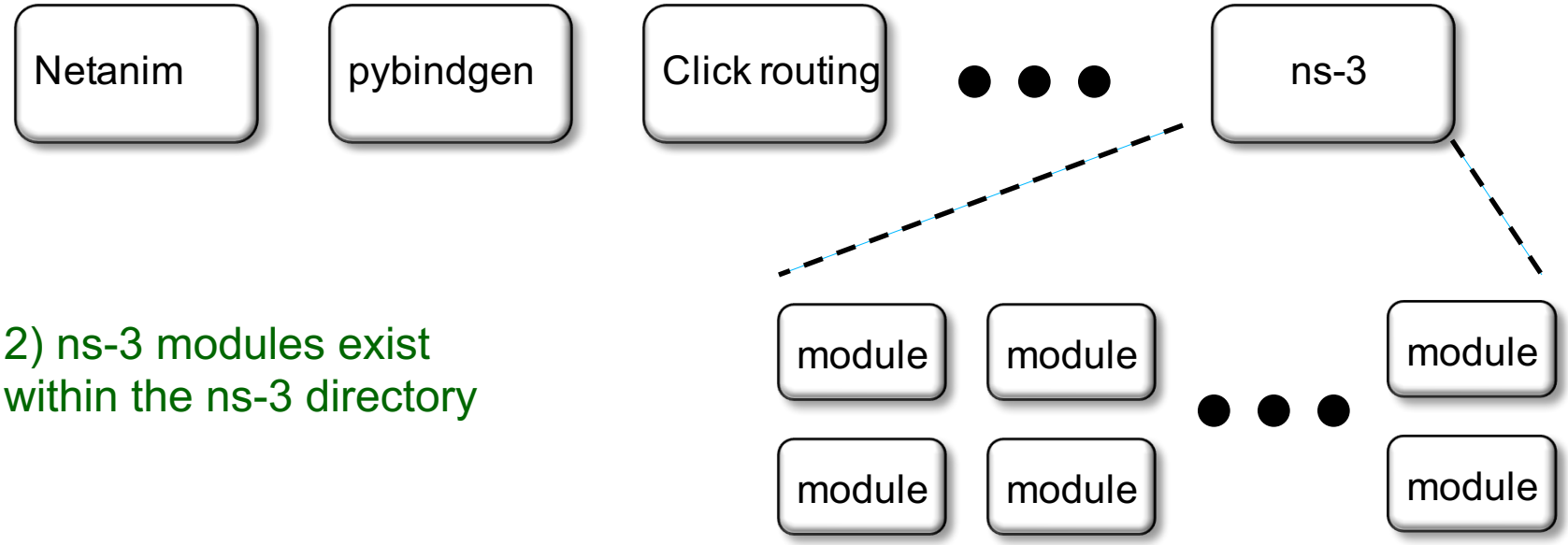
Used to measure KPIs



Software organization

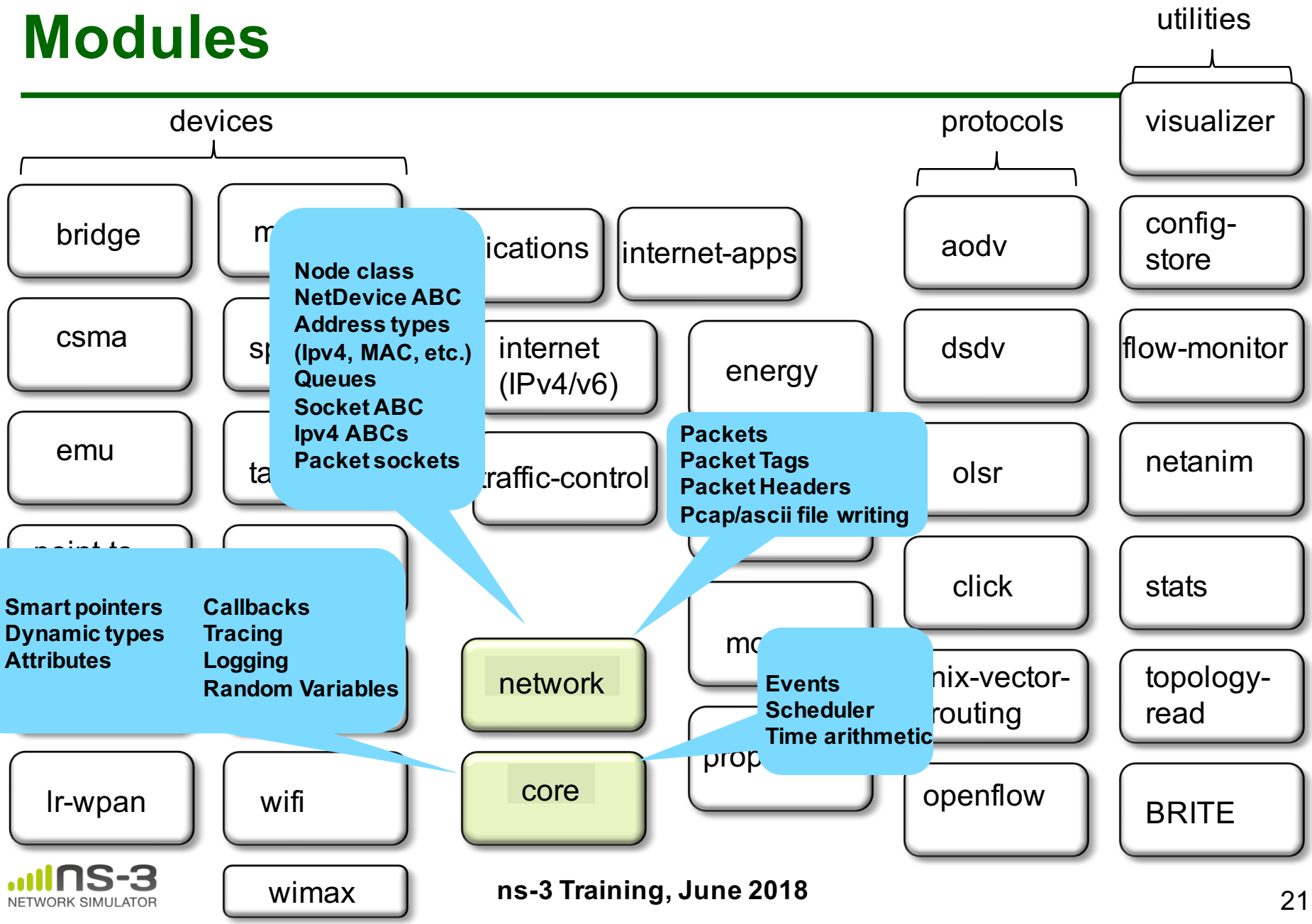
- Two levels of ns-3 software and libraries

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



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

Modules


utilities
devices
protocols
visualizer
bridge
csma
emu
lr-wpan
wifi
wimax
internet-apps
internet (IPv4/v6)
energy
aodv
dsdv
olsr
click
openflow
config-store
flow-monitor
netanim
stats
topology-read
BRITE
Node class
NetDevice ABC
Address types (Ipv4, MAC, etc.)
Queues
Socket ABC
Ipv4 ABCs
Packet sockets
Packets
Packet Tags
Packet Headers
Pcap/ascii file writing
Smart pointers
Dynamic types
Attributes
Callbacks
Tracing
Logging
Random Variables
Events
Scheduler
Time arithmetic
network
core

Module organization

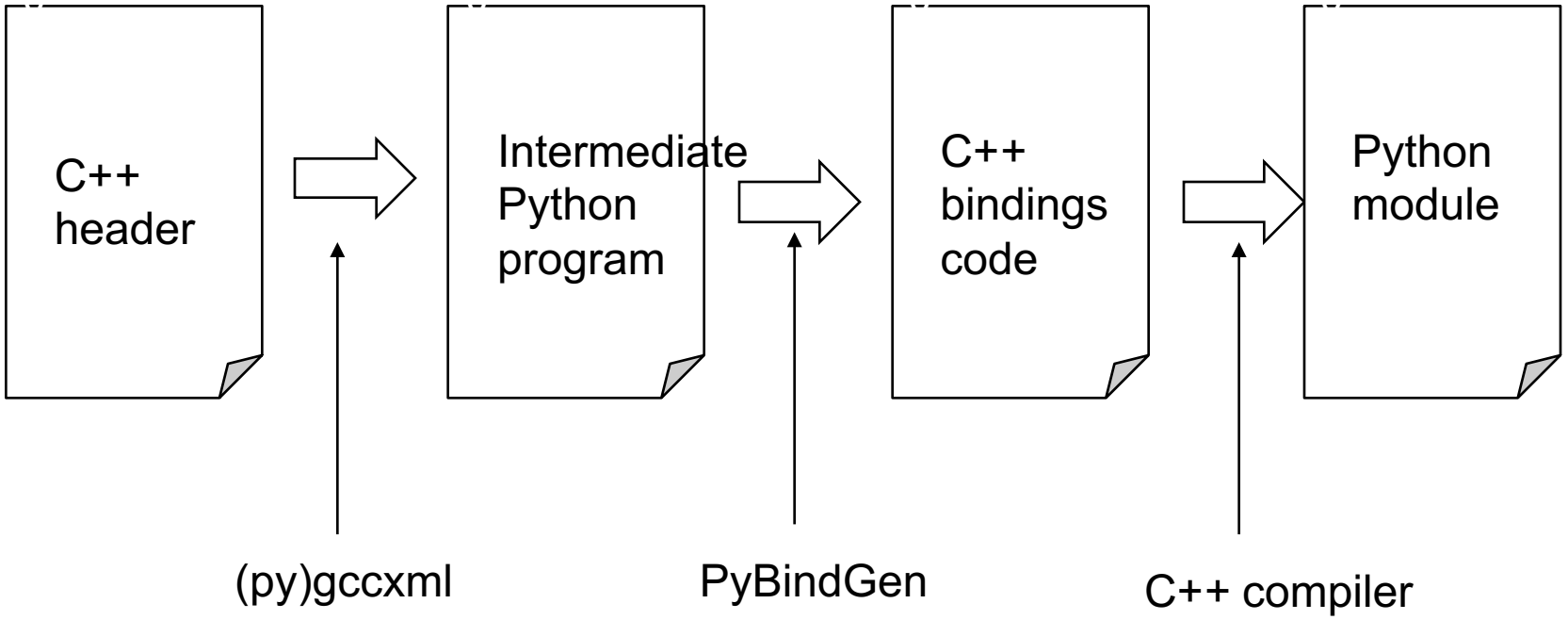
- models/
- examples/
- tests/
- bindings/
- doc/
- wscript

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



Python bindings status

- API scanning for Python used to use a tool called gccxml
- ns-3 has moved to the successor, CastXML
  - requires a development installation of clang
- Automated testing currently only for Linux 64-bit systems
  - MacOS bindings require some manual changes

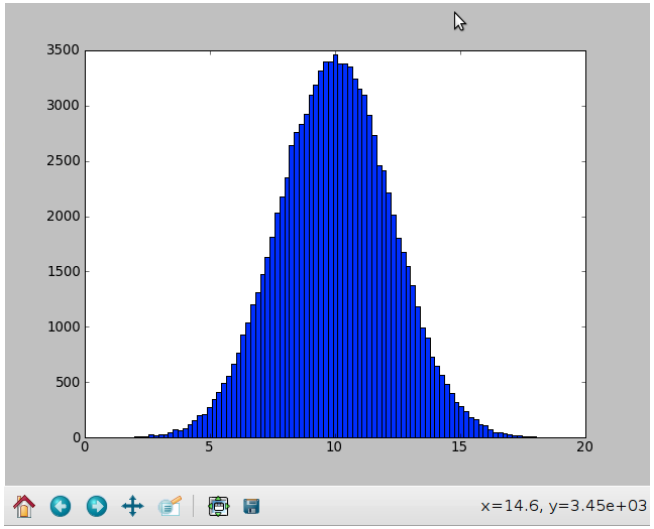
Integrating other tools and libraries

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

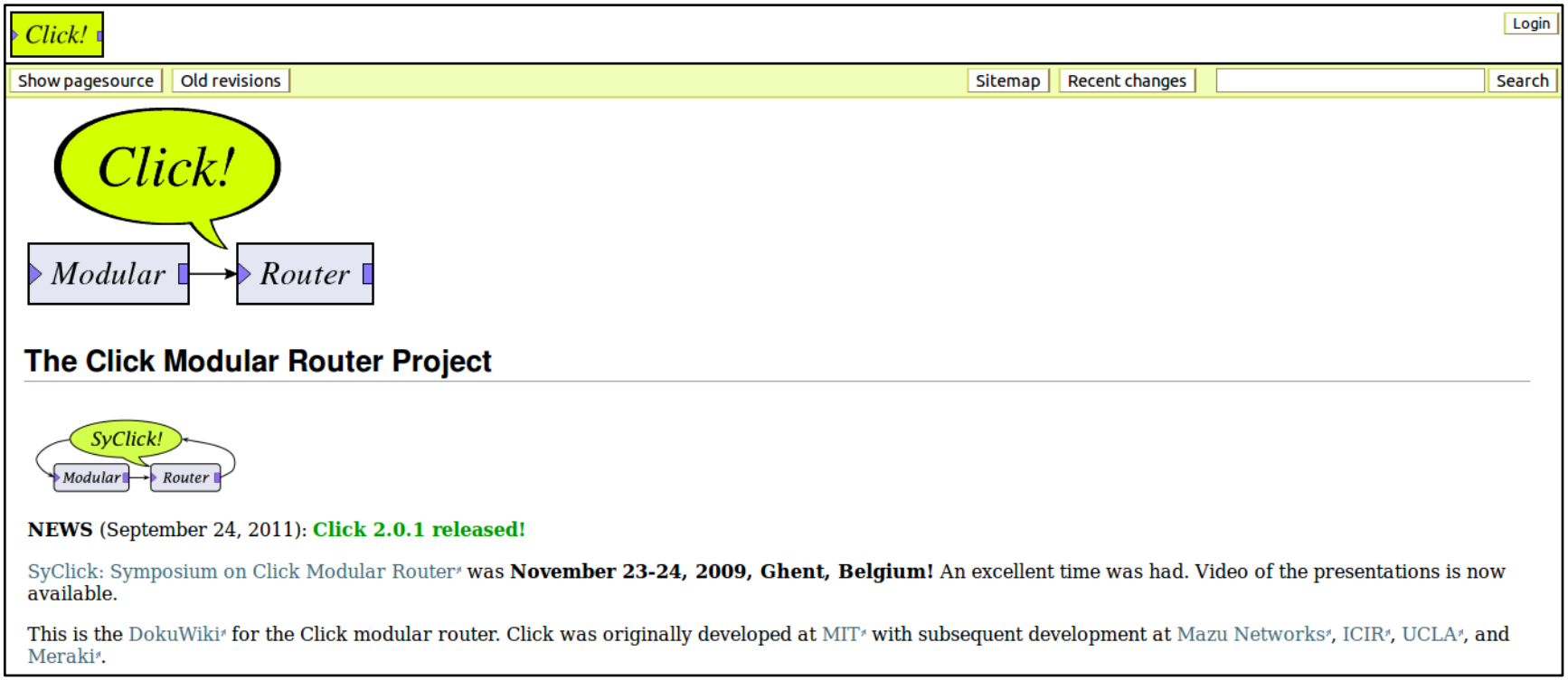
Matplotlib

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

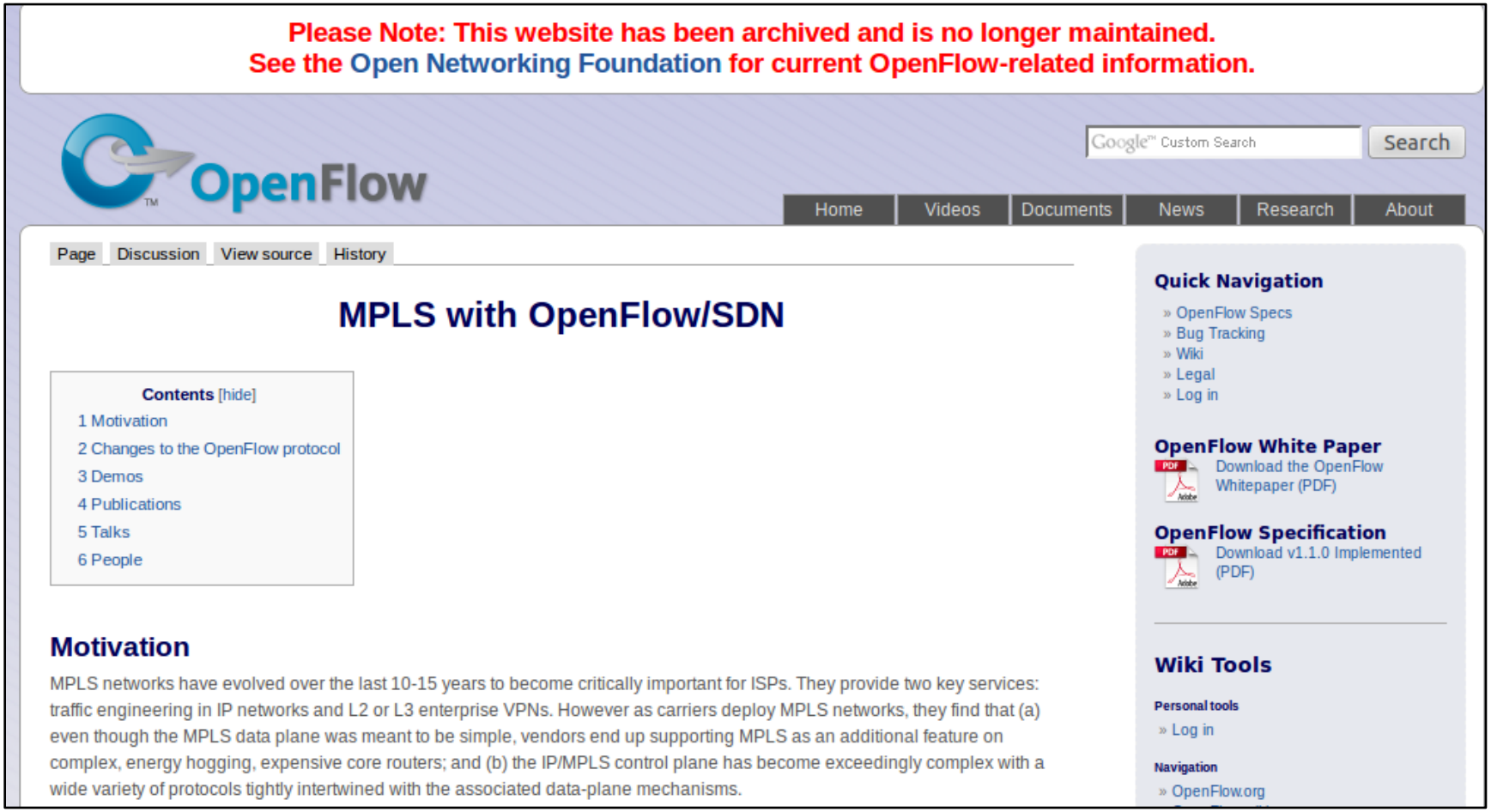



```
# Demonstrate use of ns-3 as a random number generator integrated
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial

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



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]

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

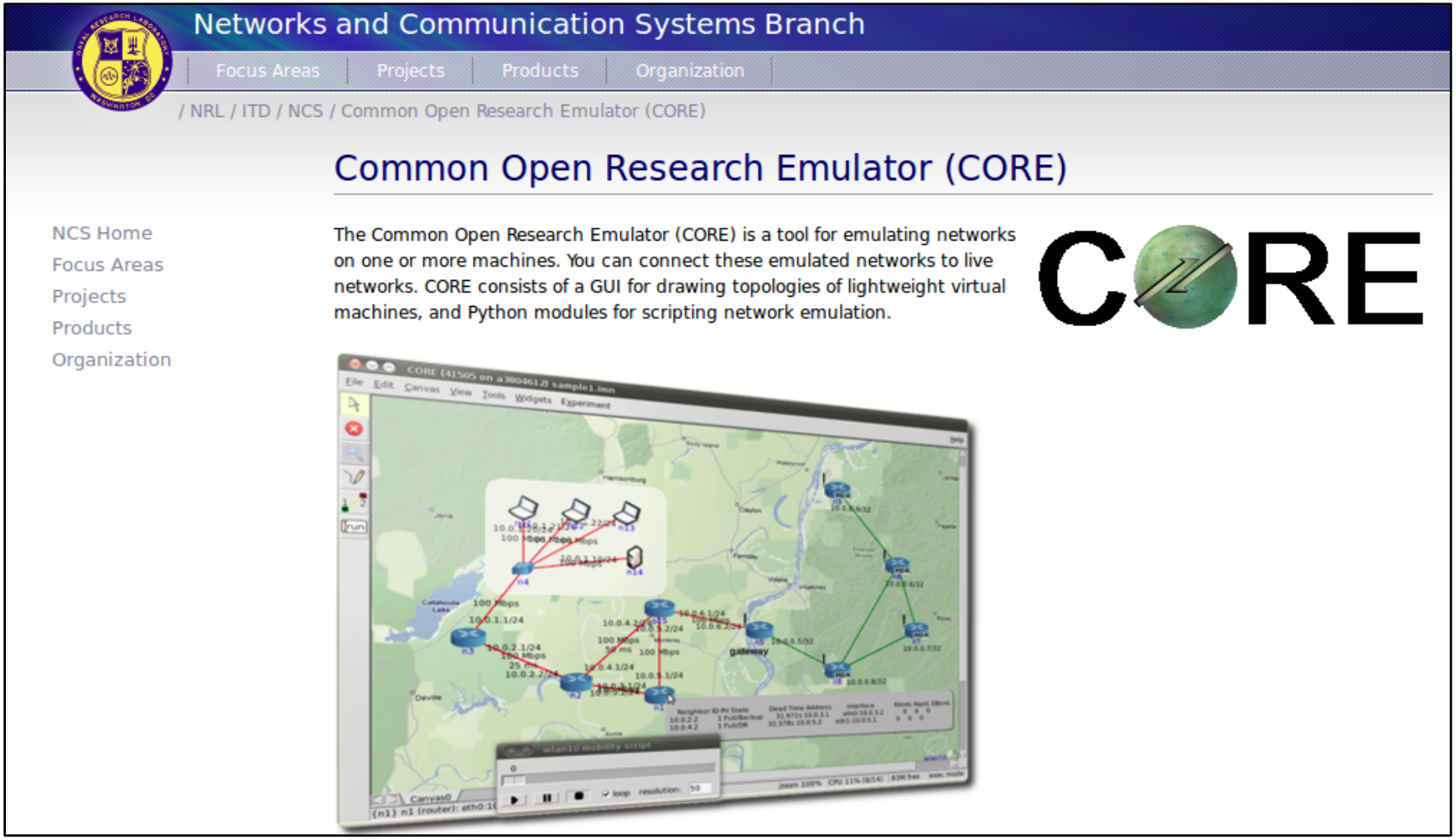
Download v1.1.0 Implemented (PDF)

Wiki Tools

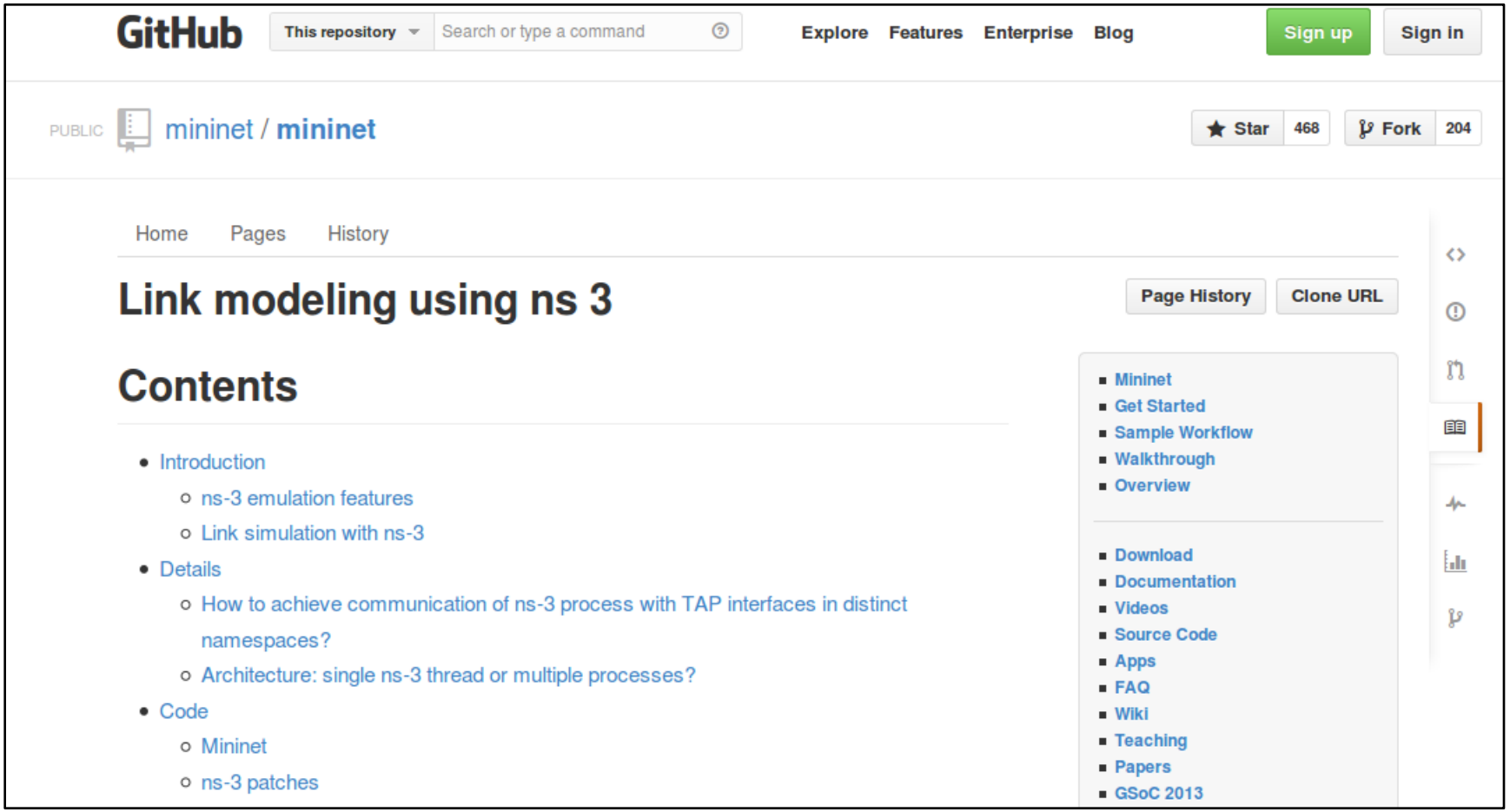
Personal tools
» Log in

Navigation
» OpenFlow.org

CORE emulator



mininet emulator



Co-simulation frameworks have emerged

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

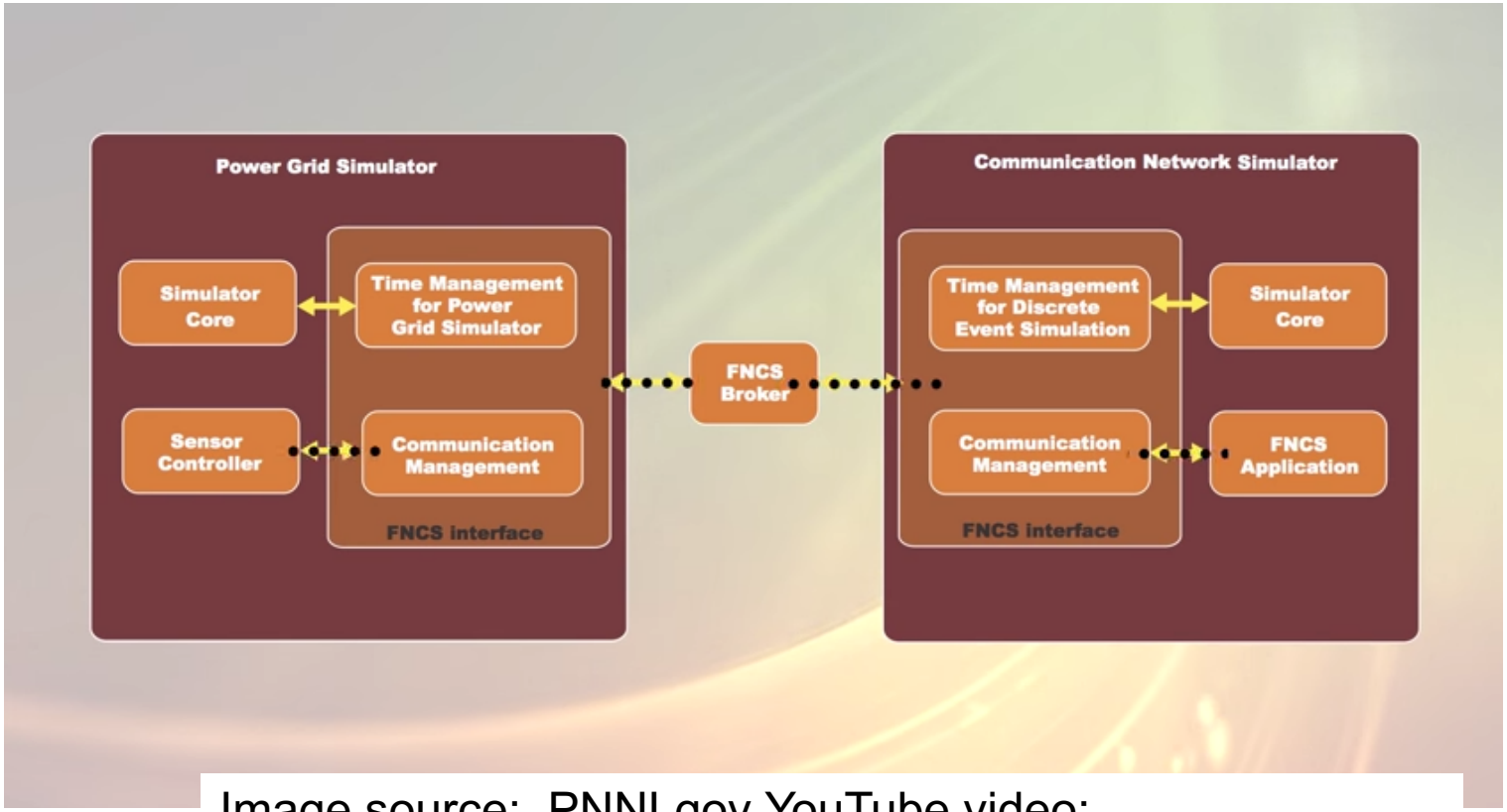


Image source: PNNLgov YouTube video:
Introducing FNCS: Framework for Network Co-Simulation

FAQs

- Does ns-3 have a Windows version?
  - A new Windows Visual Studio release is underway
  - Windows 10 'Linux subsystem' also worksDoes 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

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

Project overview

ns-3 main website

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

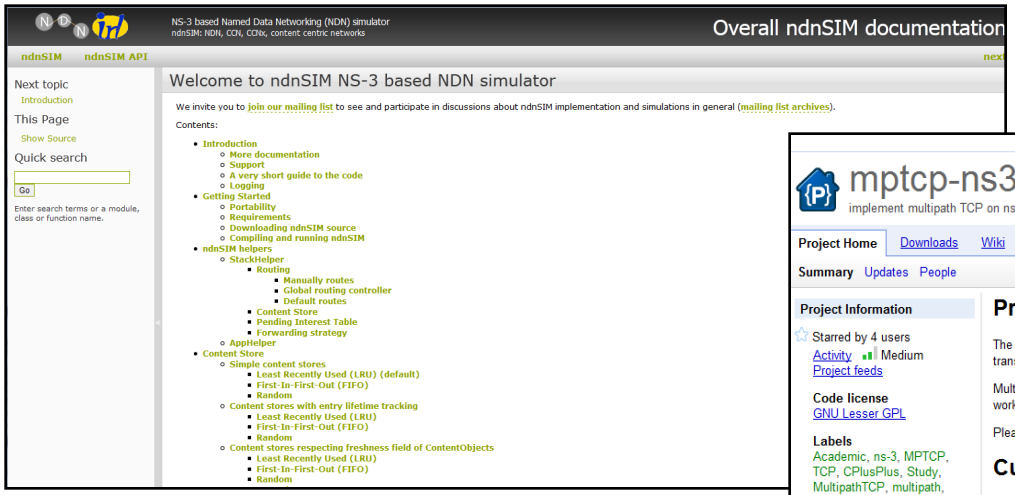
How the project operates

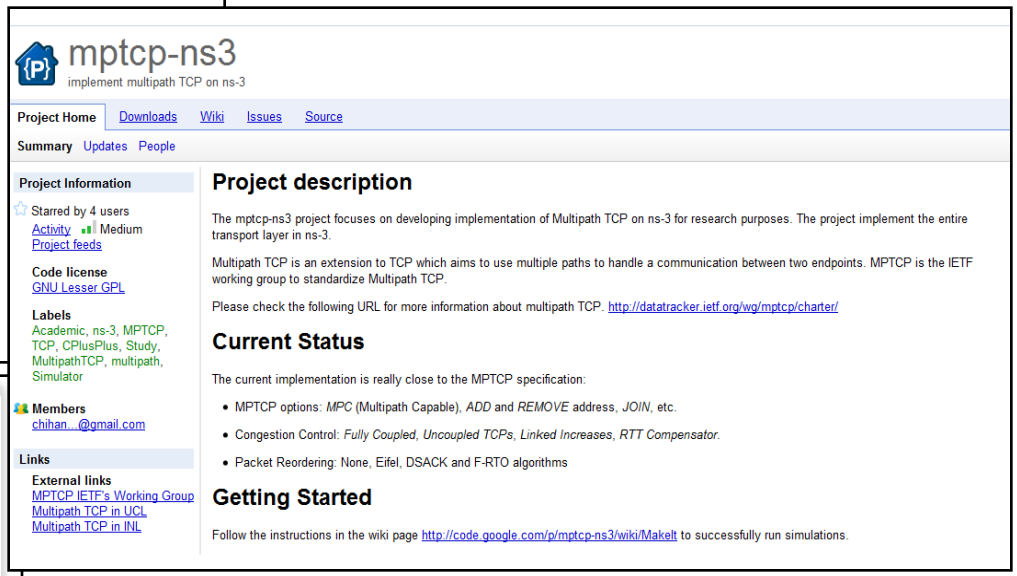
- Project provides three annual software releases
- Users interact on mailing lists and using Bugzilla bug tracker
- Code may be proposed for merge
  - Code reviews occur on a Google site
- Maintainers (one for each module) fix or delegate bugs, participate in reviews
- Project has been conducting annual 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...)

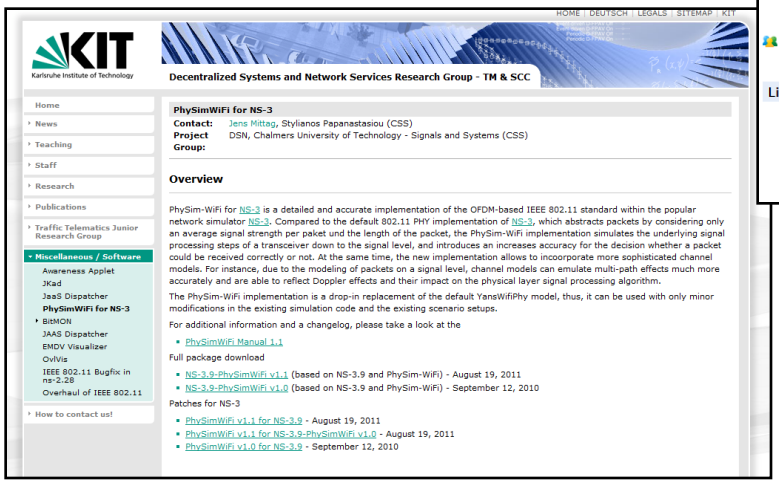
Maintainers, Authors, Users

- ~10-15 maintainers at any given time
- 228 authors credited in AUTHORS file
- Over 8000 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

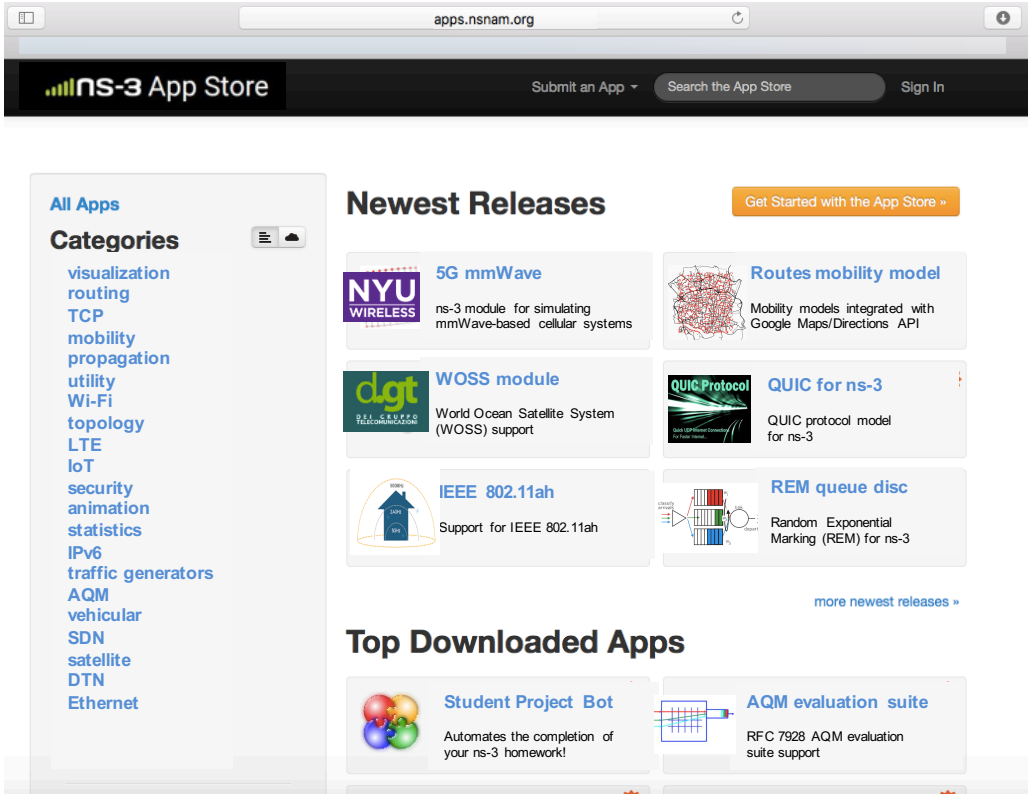






Migrating to an ns-3 app store

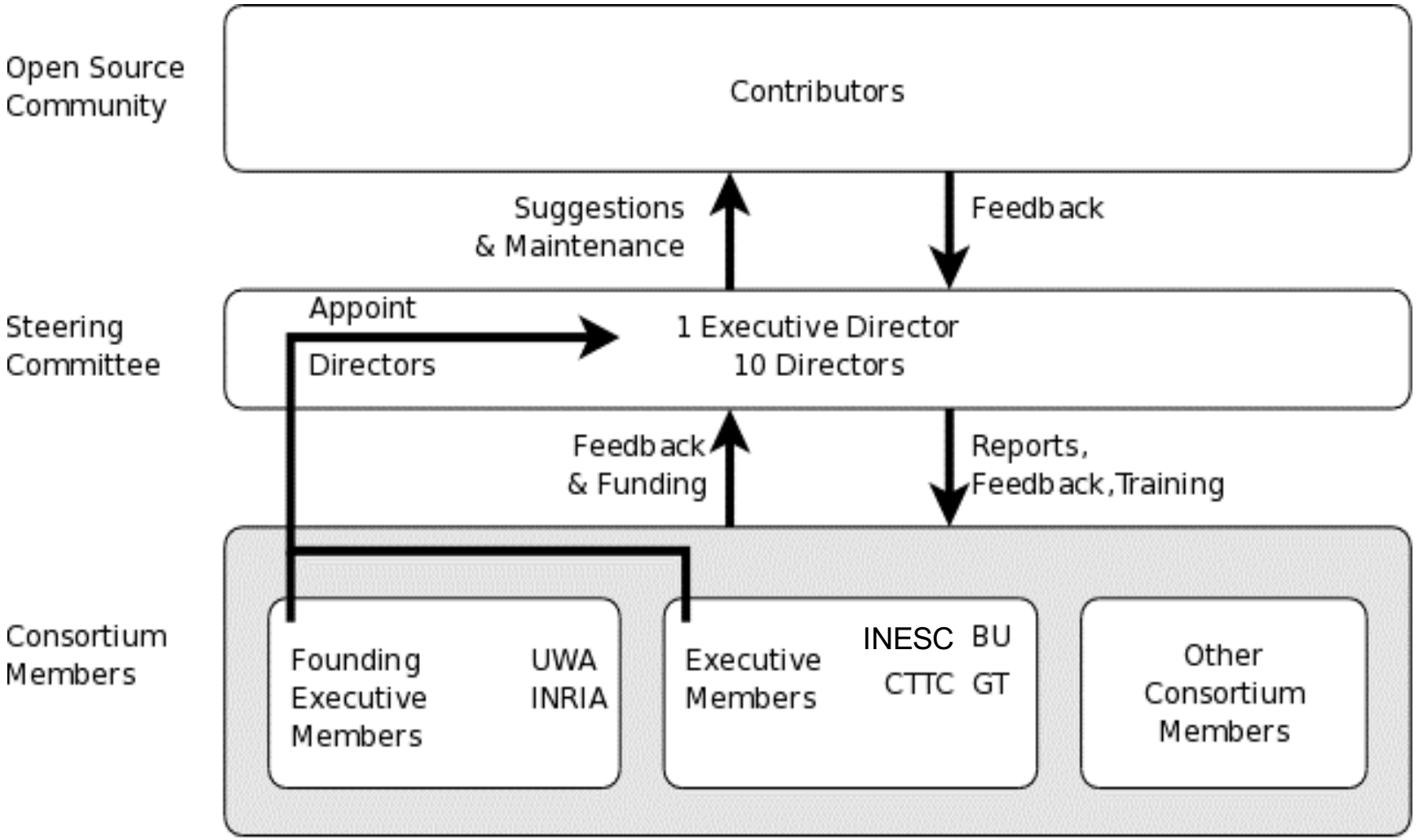
- **Developed by:** Abhijith Anilkumar, National Institute of Technology Karnataka, India.
- **Goal:** Support modularization of ns-3 codebase.
- **Beta site:** https://apps.nsnam.org



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