ns-3 Introduction

Tom Henderson (University of Washington)

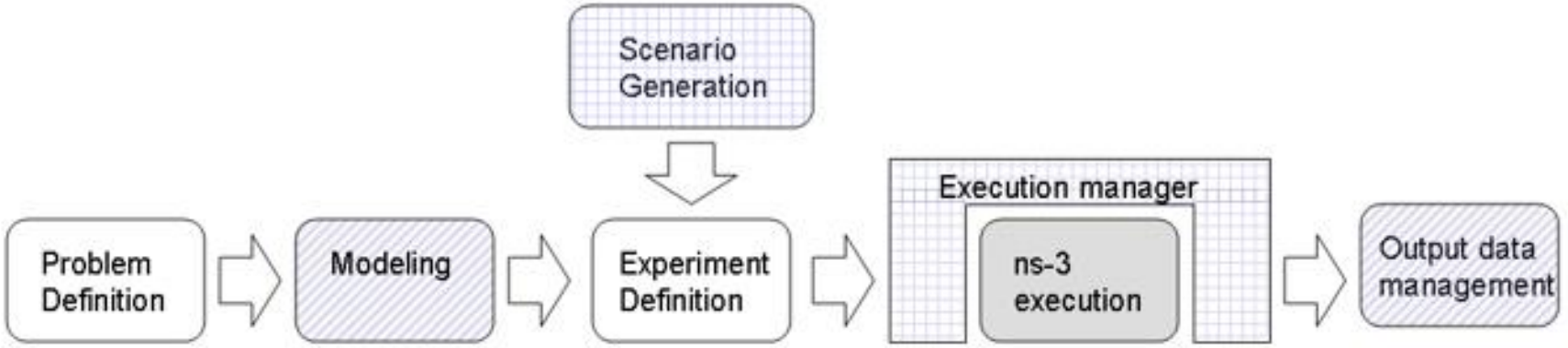
July 2014

Agenda

- ns-3 project overview
  - What is ns-3?
  - Why use ns-3?
  - Project organization
  - Relationship to ns-2
  - Future directions
- Getting started with ns-3

Discrete event network simulator

- Model of the evolution of a networked system through discrete events in time
- Used for experimentation and education



ns-3 simulation basics

- Simulation time advances 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

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

Software orientation

Key differences from other network simulators:

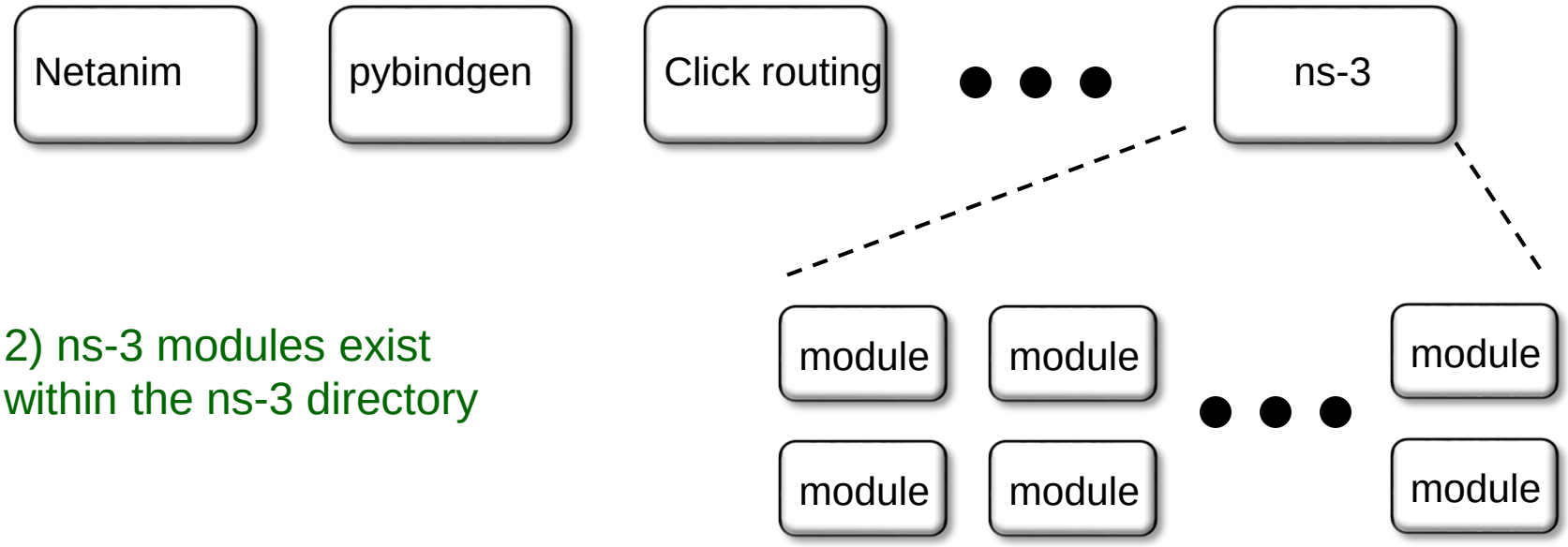
1) Command-line, Unix orientation
   - vs. Integrated Development Environment (IDE)

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

Software organization

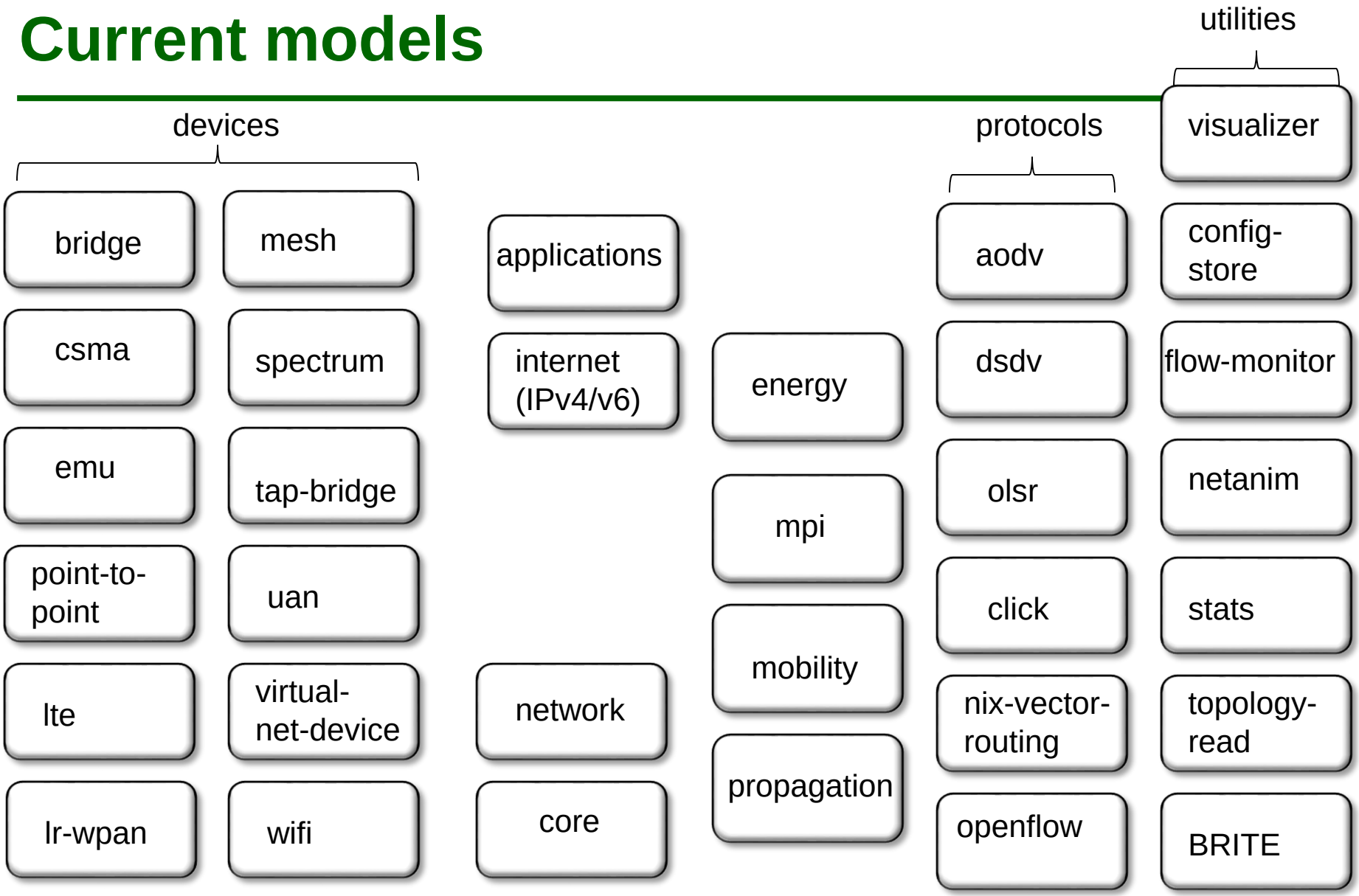
- Two levels of ns-3 software and libraries

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

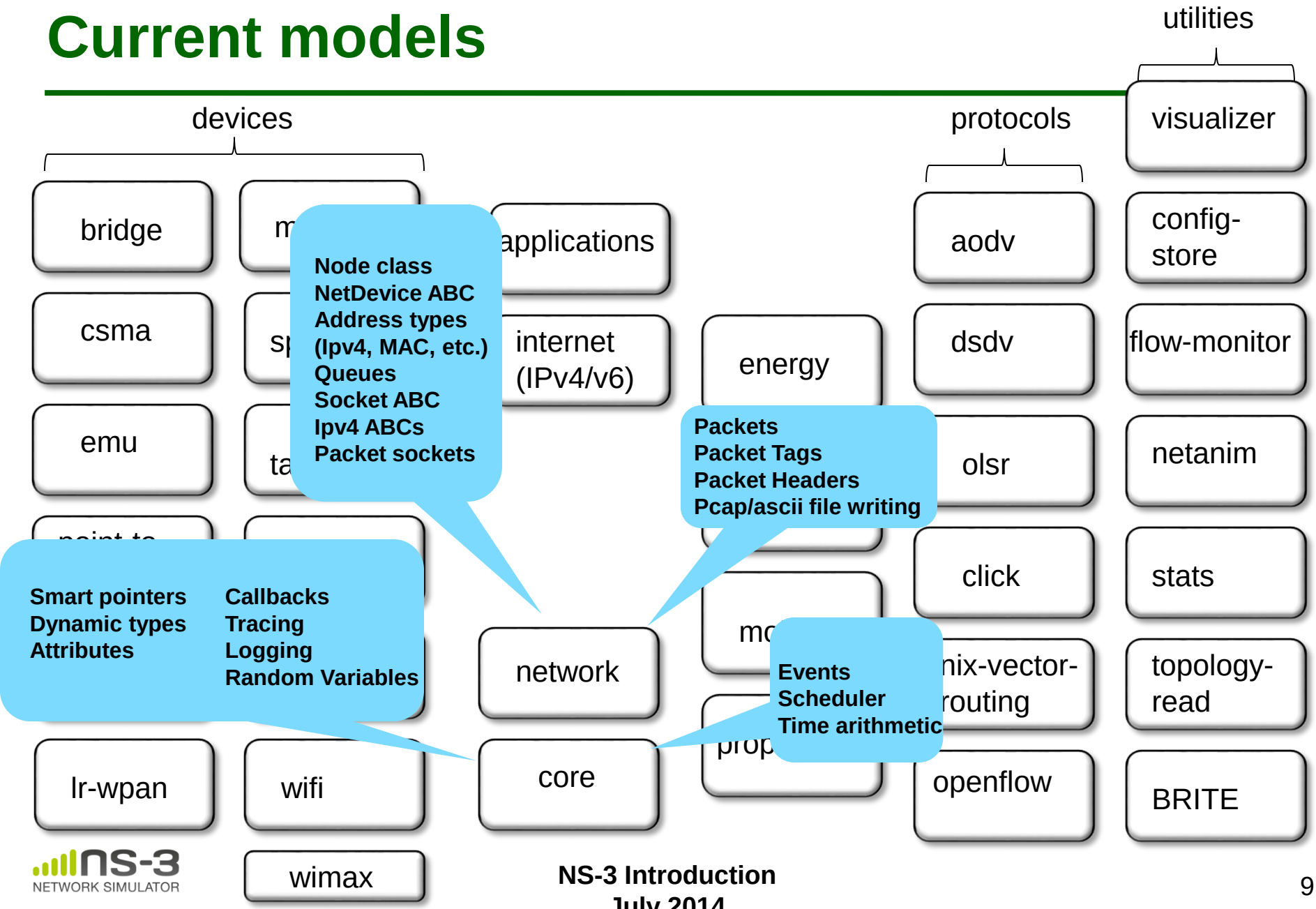


2) ns-3 modules exist within the ns-3 directory

Current models

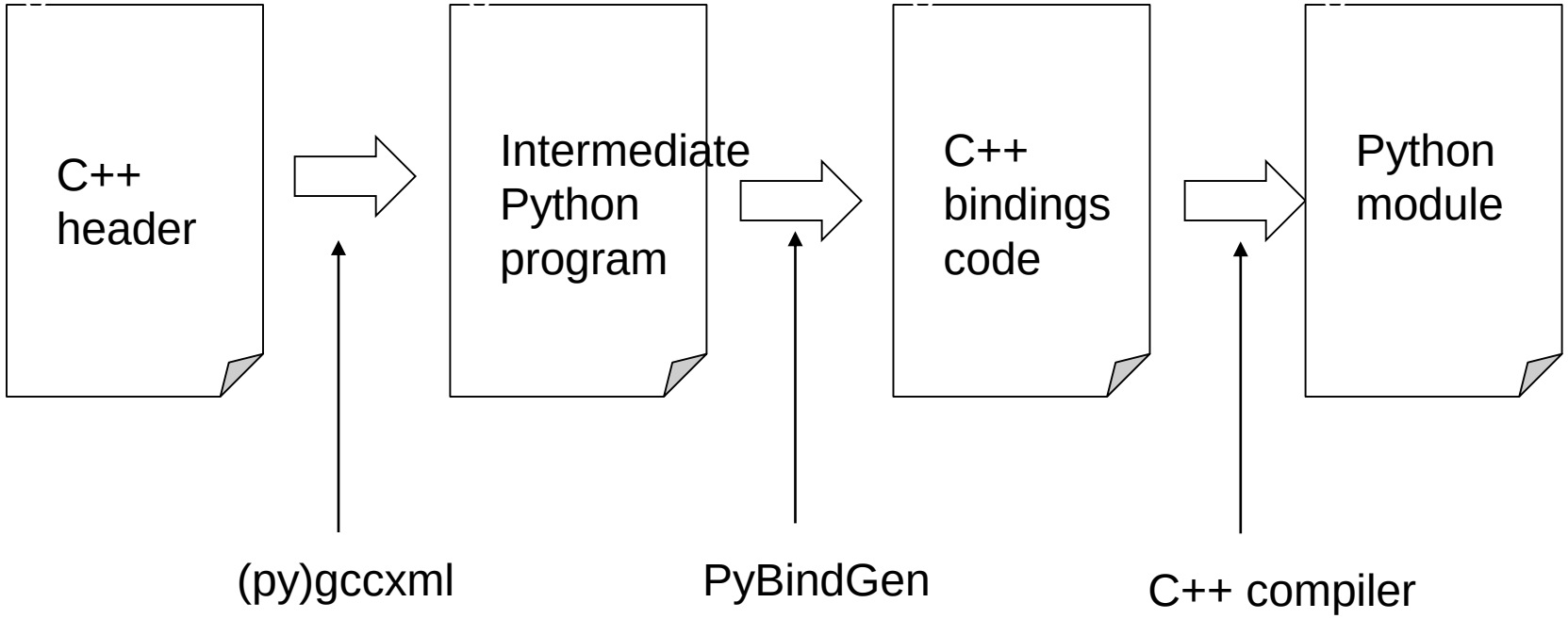


Current models



Python bindings

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



Agenda

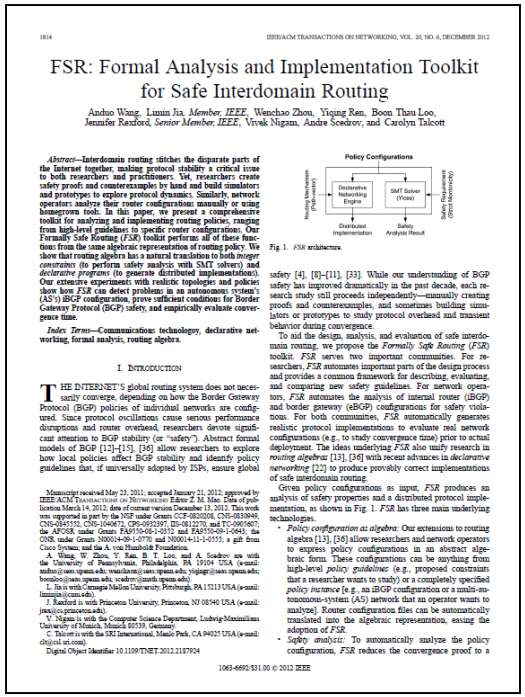
- ns-3 project overview
  - What is ns-3?
  - Why use ns-3?
  - Project organization
  - Relationship to ns-2
  - Future directions
- Getting started with ns-3

Why use ns-3?

- You want to study *network performance* or *protocol operation* in a *controllable* or *scalable* environment
- You are comfortable writing C++ or Python code, and combining ns-3 with other code
- You like the idea of working on an active open source project
- ns-3 has the models you are looking for
  - or you can provide/integrate what is lacking

What have people done with ns-3?

- ~750 publications to date
  - 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

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

Abstract—Interdomain routing stitches the disparate parts of the Internet together, making protocol stability a critical issue to both researchers and practitioners. Yet, researchers create safety proofs and counterexamples by hand and build simulators and prototypes to explore protocol dynamics. Similarly, network operators analyze their router configurations manually or using homegrown tools. In this paper, we present a comprehensive toolkit for analyzing and implementing routing policies, ranging from high-level guidelines to specific router configurations. Our Formally Safe Routing (*FSR*) toolkit performs all of these functions: from the same algebraic representation of routing policy. We show that routing algebra has a natural translation to both *integer constraints* (to perform safety analysis with SMT solvers) and *declarative programs* (to generate distributed implementations). Our extensive experiments with realistic topologies and policies show how *FSR* can detect problems in an autonomous system's (AS's) iBGP configuration, prove sufficient conditions for Border Gateway Protocol (BGP) safety, and empirically evaluate convergence time.

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

I. Introduction

THE INTERNET'S global routing system does not necessarily converge, depending on how the Border Gateway Protocol (BGP) policies of individual networks are configured. Since protocol oscillations cause serious performance disruptions and router overhead, researchers devote significant attention to BGP stability (or "safety"). Abstract formal models of BGP [12]–[15], [36] allow researchers to explore how local policies affect BGP stability and identify policy guidelines that, if universally adopted by ISPs, ensure global

Manuscript received May 23, 2011; accepted January 21, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. M. Mao. Date of publication March 14, 2012; date of current version December 13, 2012. This work was supported in part by the NSF under Grants CCF-0820208, CNS-0830949, CNS-0845552, CNS-1040672, CPS-0932397, III-0812270, and TC-0905607; the AFOSR, under Grants FA9550-08-1-0352 and FA9550-09-1-0643; the ONR, under Grants N00014-09-1-0770 and N00014-11-1-0555; a gift from Cisco System; and the A. von Humboldt Foundation.

A. Wang, W. Zhou, Y. Ren, B. T. Loo, and A. Scedrov are with the University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: andno@seas.upenn.edu; wenchaoz@seas.upenn.edu; yiqing@seas.upenn.edu; boonloo@seas.upenn.edu; scedrov@math.upenn.edu).

L. Jia is with Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: liminjia@cmu.edu).

J. Rexford is with Princeton University, Princeton, NJ 08540 USA (e-mail: jrex@cs.princeton.edu).

V. Nigam is with the Computer Science Department, Ludwig-Maximilians University of Munich, Munich 80539, Germany.

C. Talcott is with the SRI International, Menlo Park, CA 94025 USA (e-mail: clt@csl.sri.com).

Digital Object Identifier 10.1109/TNET.2012.2187924

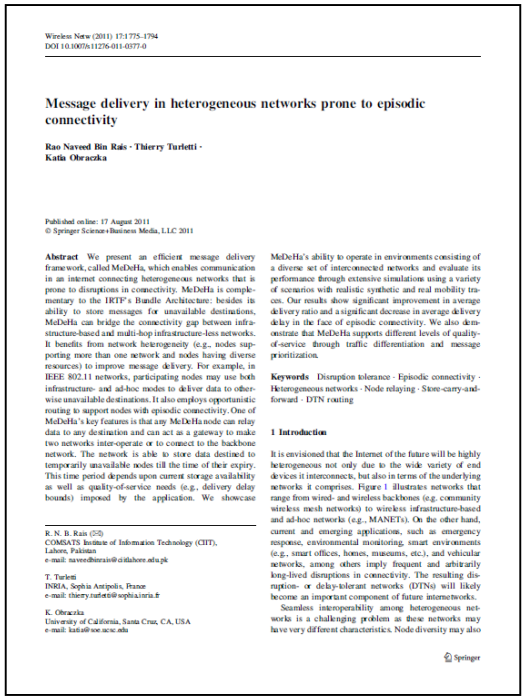
Fig. 1. *FSR* architecture.

safety [4], [8]–[11], [33]. While our understanding of BGP safety has improved dramatically in the past decade, each research study still proceeds independently—manually creating proofs and counterexamples, and sometimes building simulators or prototypes to study protocol overhead and transient behavior during convergence.

To aid the design, analysis, and evaluation of safe interdomain routing, we propose the *Formally Safe Routing* (*FSR*) toolkit. *FSR* serves two important communities. For researchers, *FSR* automates important parts of the design process and provides a common framework for describing, evaluating, and comparing new safety guidelines. For network operators, *FSR* automates the analysis of internal router (iBGP) and border gateway (eBGP) configurations for safety violations. For both communities, *FSR* automatically generates realistic protocol implementations to evaluate real network configurations (e.g., to study convergence time) prior to actual deployment. The ideas underlying *FSR* also unify research in *routing algebras* [13], [36] with recent advances in *declarative networking* [22] to produce provably correct implementations of safe interdomain routing.

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

- *Policy configuration as algebra:* Our extensions to routing algebra [13], [36] allow researchers 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 researcher wants to study) or a completely specified *policy instance* [e.g., an iBGP configuration or a multi-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



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

Published online: 17 August 2011

Abstract We present an efficient message delivery framework, called MeDeHa, which enables communication in an internet connecting heterogeneous networks that is prone to disruptions in connectivity. MeDeHa is complementary to the IRTF's Bundle Architecture: besides its ability to store messages for unavailable destinations, MeDeHa can bridge the connectivity gap between infrastructure-based and multi-hop infrastructure-less networks. It benefits from network heterogeneity (e.g., nodes supporting more than one network and nodes having diverse resources) to improve message delivery. For example, in IEEE 802.11 networks, participating nodes may use both infrastructure- and ad-hoc modes to deliver data to otherwise unavailable destinations. It also employs opportunistic routing to support nodes with episodic connectivity. One of MeDeHa's key features is that any MeDeHa node can relay data to any destination and can act as a gateway 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 MeDeHa'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 scenarios with realistic synthetic and real mobility traces. Our results show significant improvement in average delivery ratio and a significant decrease in average delivery delay in the face of episodic connectivity. We also demonstrate that MeDeHa supports different levels of quality-of-service through traffic differentiation and message prioritization.

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

1 Introduction

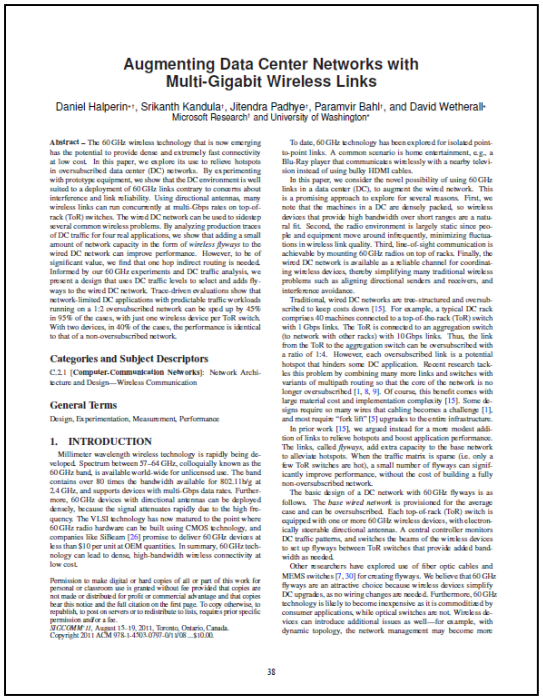
It is envisioned that the Internet of the future will be highly heterogeneous not only due to the wide variety of end devices it interconnects, but also in terms of the underlying networks it comprises. Figure 1 illustrates networks that range from wired- and wireless backbones (e.g. community wireless 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, museums, etc.), and vehicular networks, among others imply frequent and arbitrarily 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: naveedbinrais@ciitlahore.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: katia@soe.ucsc.edu



Augmenting Data Center Networks with Multi-Gigabit Wireless Links

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

Abstract – The 60 GHz wireless technology that is now emerging has the potential to provide dense and extremely fast connectivity at low cost. In this paper, we explore its use to relieve hotspots in oversubscribed data center (DC) networks. By experimenting with prototype equipment, we show that the DC environment is well suited to a deployment of 60 GHz links contrary to concerns about interference and link reliability. Using directional antennas, many wireless links can run concurrently at multi-Gbps rates on top-of-rack (ToR) switches. The wired DC network can be used to sidestep several common wireless problems. By analyzing production traces of DC traffic for four real applications, we show that adding a small amount of network capacity in the form of *wireless flyways* to the wired DC network can improve performance. However, to be of significant value, we find that one hop indirect routing is needed. Informed by our 60 GHz experiments and DC traffic analysis, we present a design that uses DC traffic levels to select and adds flyways to the wired DC network. Trace-driven evaluations show that network-limited DC applications with predictable traffic workloads running on a 1:2 oversubscribed network can be sped up by 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-oversubscribed 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 world-wide for unlicensed use. The band contains over 80 times the bandwidth available for 802.11b/g at 2.4 GHz, and supports devices with multi-Gbps data rates. Furthermore, 60 GHz devices with directional antennas can be deployed densely, because the signal attenuates rapidly due to the high frequency. The VLSI technology has now matured to the point where 60 GHz radio hardware can be built using CMOS technology, and companies like Silkam [26] promise to deliver 60 GHz devices at less than $10 per unit at OEM quantities. In summary, 60 GHz technology can lead to dense, high-bandwidth wireless connectivity at low cost.

SIGCOMM'11, August 15–19, 2011, Toronto, Ontario, Canada.

To date, 60 GHz technology has been explored for isolated point-to-point links. A common scenario is home entertainment, e.g., a Blu-Ray player that communicates wirelessly with a nearby television instead of using bulky HDMI cables.

In this paper, we consider the novel possibility of using 60 GHz links in a data center (DC), to augment the wired network. This is a promising approach to explore for several reasons. First, we note that the machines in a DC are densely packed, so wireless devices that provide high bandwidth over short ranges are a natural fit. Second, the radio environment is largely static since people and equipment move around infrequently, minimizing fluctuations in wireless link quality. Third, line-of-sight communication is achievable by mounting 60 GHz radios on top of racks. Finally, the wired DC network is available as a reliable channel for coordinating wireless devices, thereby simplifying many traditional wireless problems such as aligning directional senders and receivers, and interference avoidance.

Traditional, wired DC networks are tree-structured and oversubscribed to keep costs down [15]. For example, a typical DC rack comprises 40 machines connected to a top-of-the-rack (ToR) switch with 1 Gbps links. The ToR is connected to an aggregation switch (to network with other racks) with 10 Gbps links. Thus, the link from the ToR to the aggregation switch can be oversubscribed with a ratio of 1:4. However, each oversubscribed link is a potential hotspot that hinders some DC application. Recent research tackles this problem by combining many more links and switches with variants of multipath routing so that the core of the network is no longer oversubscribed [1, 8, 9]. Of course, this benefit comes with large material cost and implementation complexity [13]. Some designs require so many wires that cabling becomes a challenge [1], and most require "fork lift" [5] upgrades to the entire infrastructure.

In prior work [15], we argued instead 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 to alleviate hotspots. When the traffic matrix is sparse (i.e. only a few ToR switches are hot), a small number of flyways can significantly improve performance, without the cost of building a fully non-oversubscribed network.

The basic design of a DC network with 60 GHz flyways is as follows. The *base wired network* is provisioned for the average case and can be oversubscribed. Each top-of-rack (ToR) switch is equipped with one or more 60 GHz wireless devices, with electronically steerable directional antennas. A central controller monitors DC traffic patterns, and switches the beams of the wireless devices to set up flyways between ToR switches that provide added bandwidth as needed.

Other researchers have explored use of fiber optic cables and MEMS switches [7, 30] for creating flyways. We believe that 60 GHz flyways are an attractive choice because wireless devices simplify DC upgrades, as no wiring changes are needed. Furthermore, 60 GHz technology is likely to become inexpensive as it is commoditized 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

Examples of recent publications

- L. Salameh et al., "HACK: Hierarchical ACKs for Efficient Wireless Medium Utilization," *Proceedings of 2014 Usenix Annual Technical Conference* (best paper award winner), June 2014.
- A. Azgin et al., "Mobility Study for Named Data Networking in Wireless Access Networks," *Proceedings of IEEE ICC 2014*, June 2014.
- Wong S.-H and Gary Chan, "Topology Optimization for Wireless Mesh with Directional Antennas," *Proceedings of IEEE ICC 2014*, June 2014.
- C. Gouveia et al., "Development and implementation of Portuguese smart distribution system," *Electric Power Systems Research, Elsevier*, vol. 116, June 2014.
- M. Alharthi et al., "An Acumen/NS-3 integration for modeling networked Cyber-Physical Systems," *2014 Biennial Symposium on Communications (QSBC)*, June 2014.
- L. Ciarletta et al., "Simulation and platform tools to develop safe flock of UAVs: a CPS application-driven research," *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014.

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

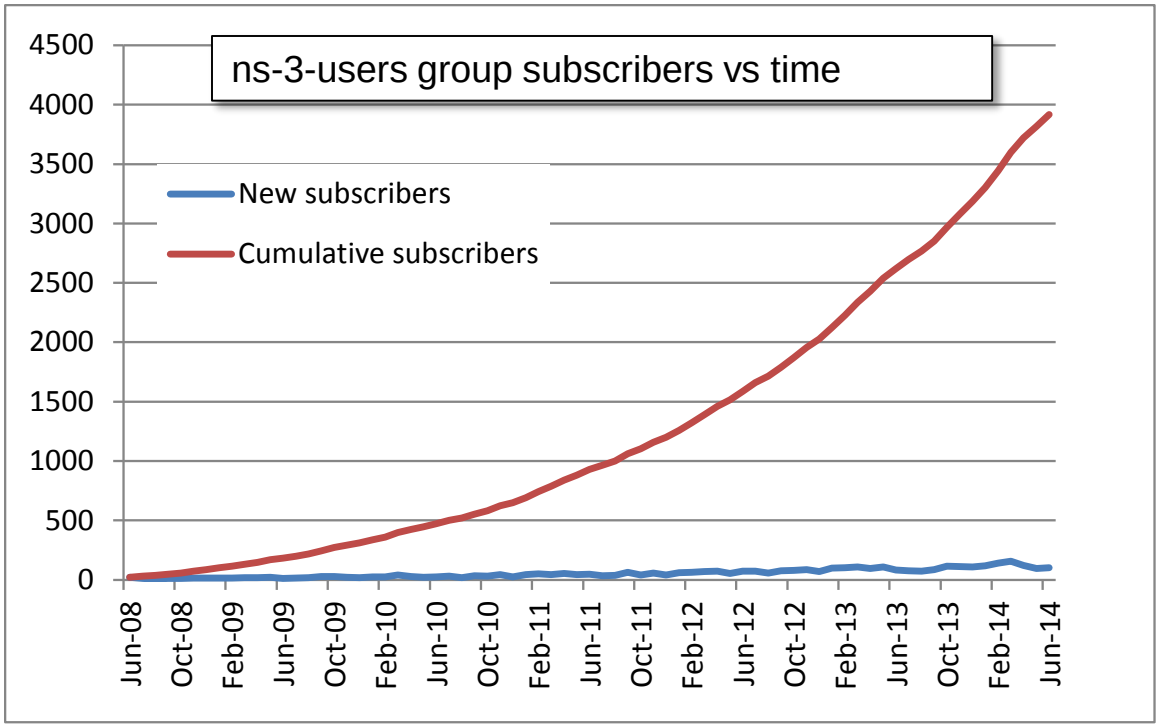
- Georgia Tech. ECE 6110 Dr. George Riley, Spring 2013 (also Fall 2011, Fall 2010)
- The University of Kansas EECS 780, EECS 882, and EECS 983 Dr. James Sterbenz, 2010 – 2012
- UPenn CIS 553/TCOM 512 Dr. Boon Thau Loo, Fall 2010
- Aalto University Jose Costa-Requena and Markus Peuhkuri, Fall 2011
- Indian 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

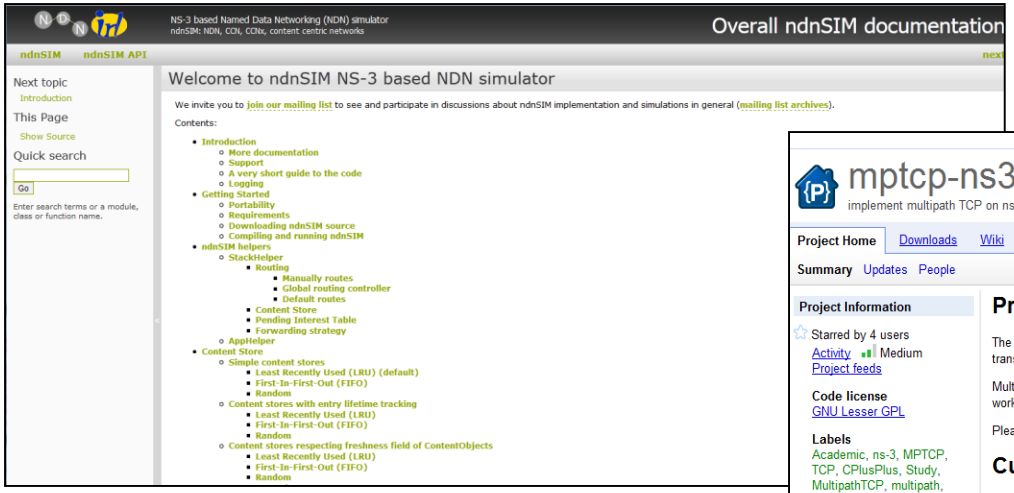
- Lalith Suresh's Lab Assignments using ns-3 page.

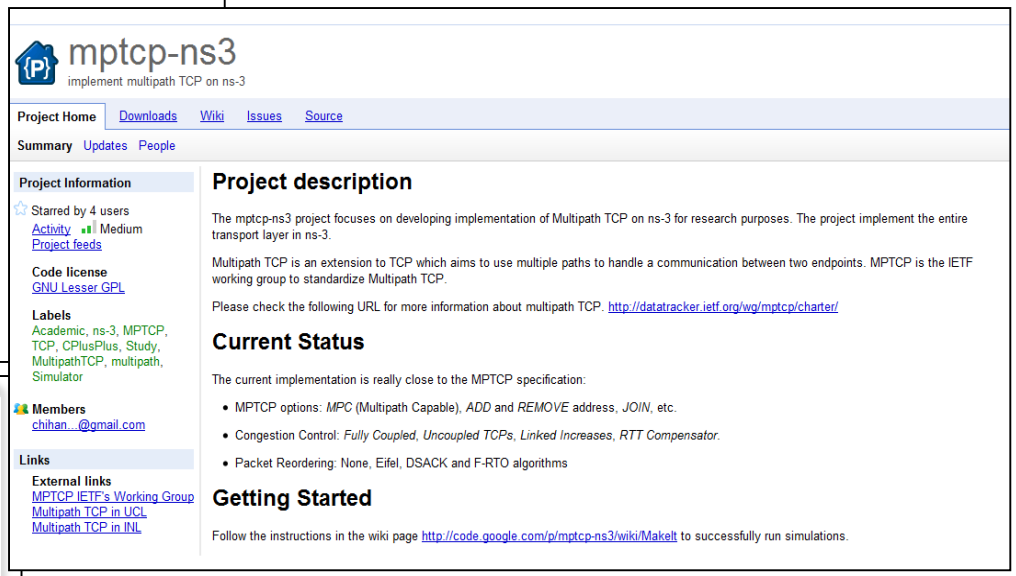
Statistics (July 2014)

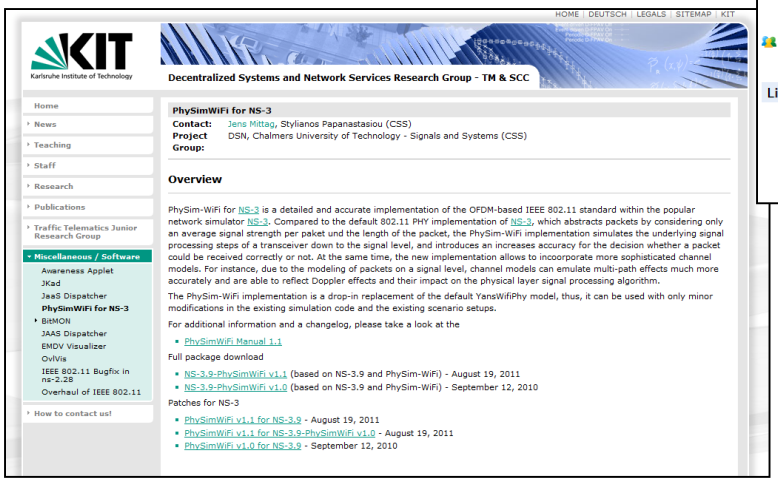
- 3900 subscribers to ns-3-users
- 1440 subscribers to ns-developers
- ~ 15 maintainers
- ~ 150 authors/contributors



Contributed code and associated projects

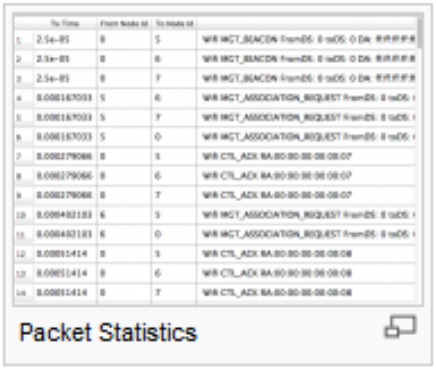


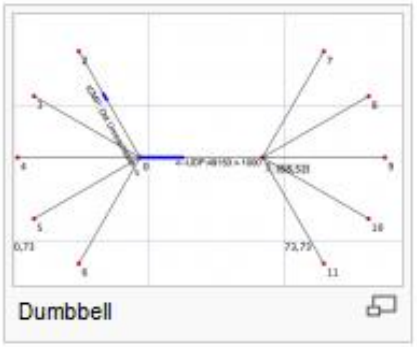


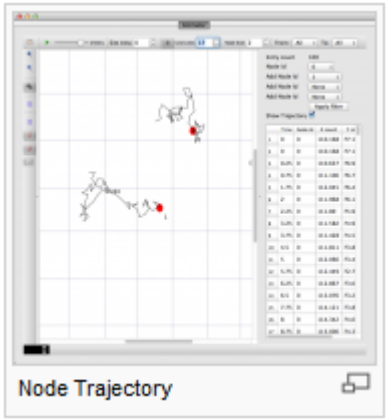


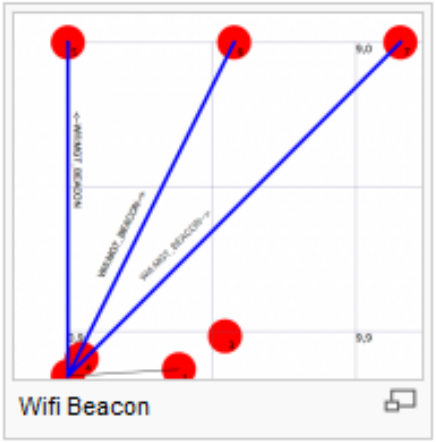
NetAnim

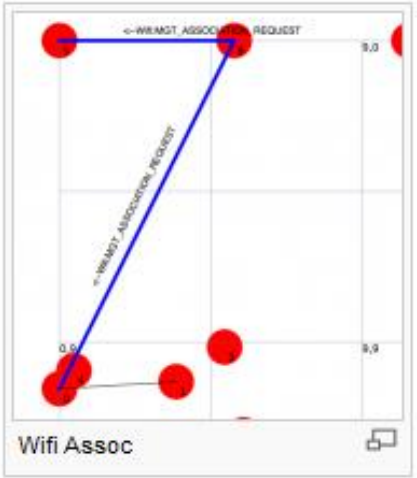
- "NetAnim" by George Riley and John Abraham
  - see the 'ns3share' channel on YouTube

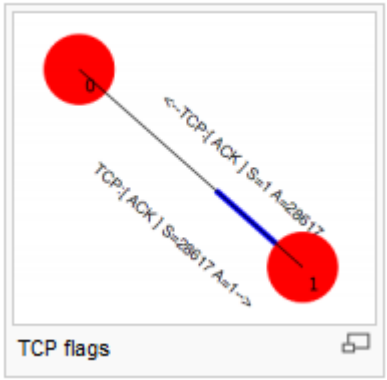








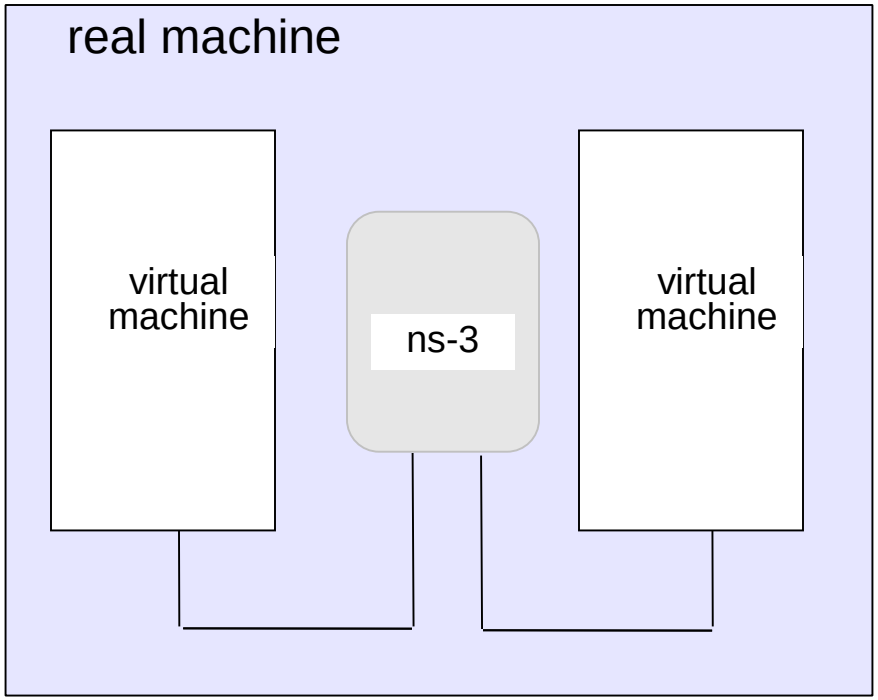


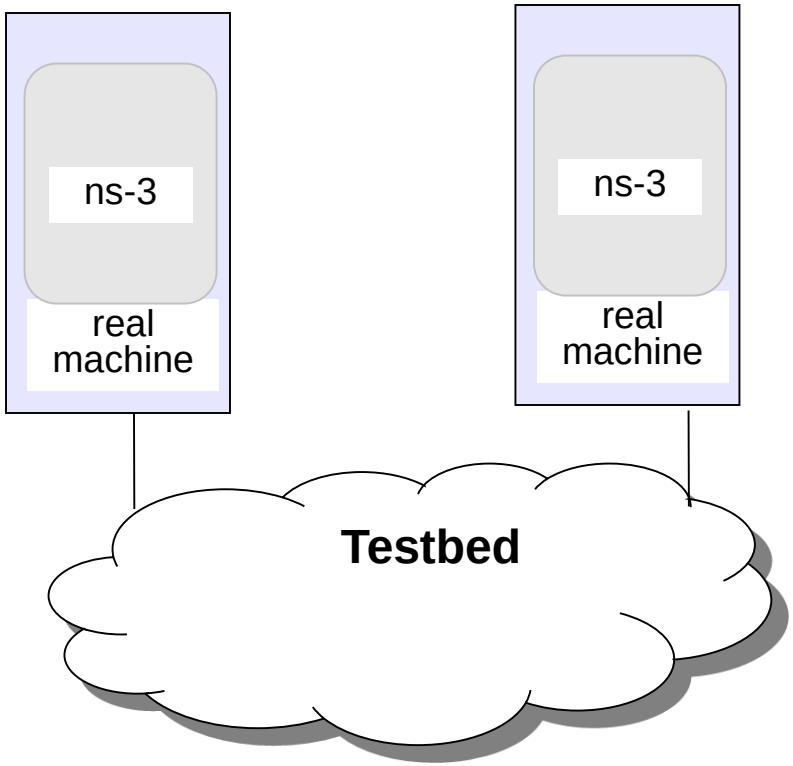


Emulation support

- Support moving between simulation and testbeds or live systems
- A real-time scheduler, and support for two modes of emulation
- Linux is only operating system supported
- Must run simulator in real time
  - `GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealTimeSimulatorImpl"));`
- Must enable checksum calculations across models
  - `GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));`
- Must sometimes run as root

ns-3 emulation modes





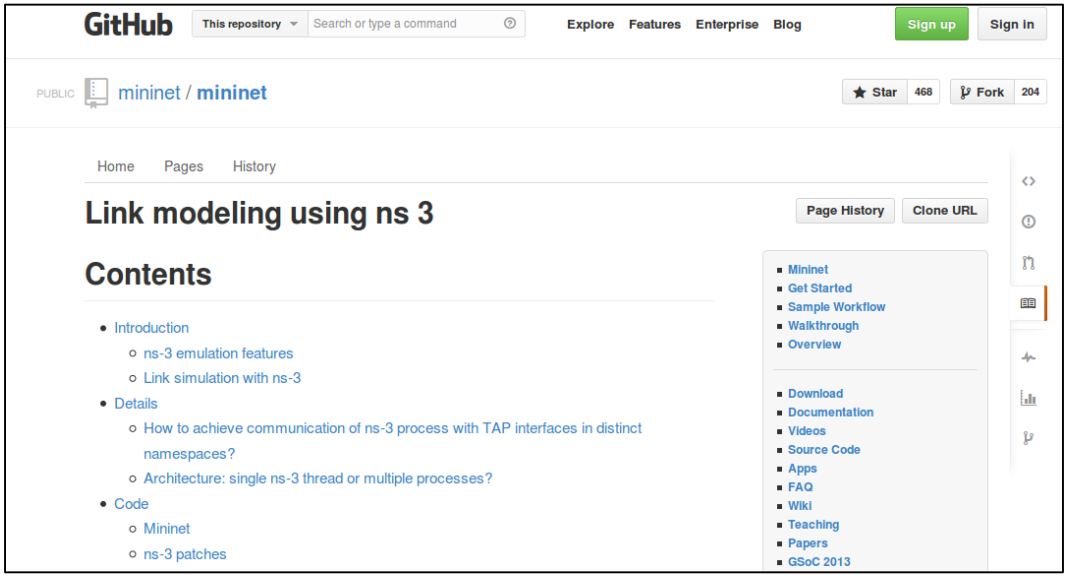
1) ns-3 interconnects real or virtual machines

2) testbeds interconnect ns-3 stacks

Various hybrids of the above are possible

Example use case: mininet

- Mininet is popular in the Software-Defined Networking (SDN) community
- Mininet uses "TapBridge" integration
- https://github.com/mininet/mininet/wiki/Link-modeling-using-ns-3

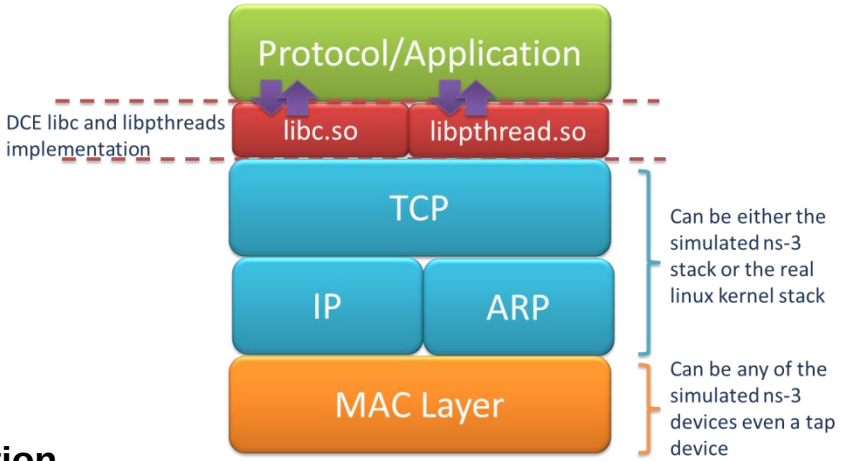


Direct Code Execution

- Lightweight virtualization of kernel and application processes, interconnected by simulated networks
- Benefits:
  - Implementation realism in controlled topologies or wireless environments
  - Model availability
  - Debugging a whole network within a single process
- Limitations:
  - Not as scalable as pure simulation
  - Tracing more limited
  - Configuration different

Direct Code Execution implementation

- 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*



DCE modes

- DCE modes in context of possible approaches

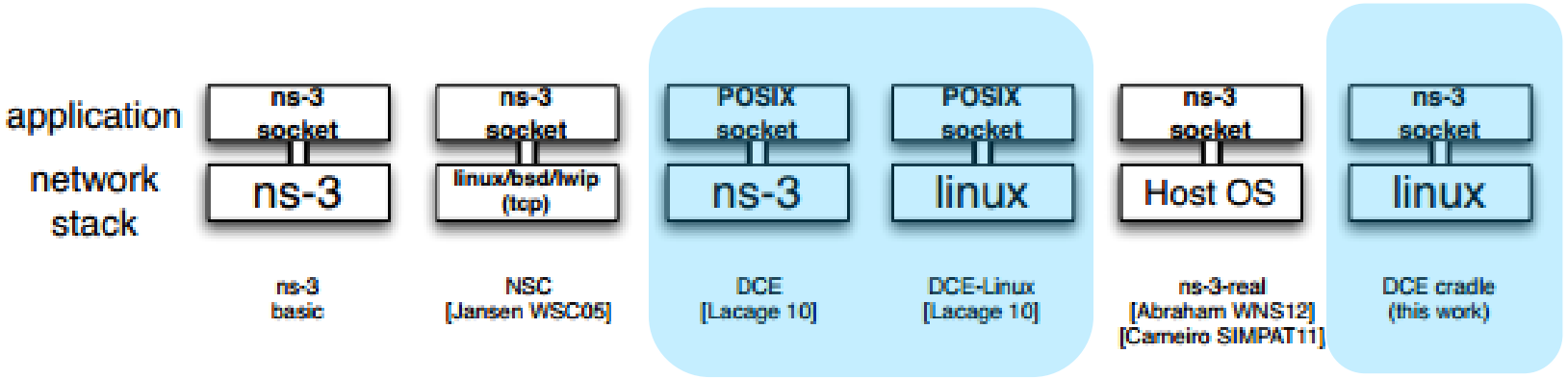


Figure 1: Current possible combinations of network stacks and applications.

Figure source: DCE Cradle: Simulate Network Protocols with Real Stacks for Better Realism, Tazaki et al, WNS3 2013.

Agenda

- ns-3 project overview
  - What is ns-3?
  - Why use ns-3?
  - Project organization
  - Relationship to ns-2
  - Future directions
- Getting started with ns-3

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

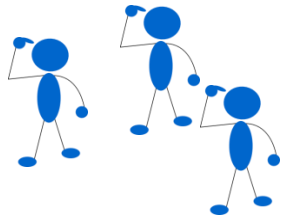
3) an open-source project that encourages community contribution, peer review, and long-term maintenance and validation of the software

How the project operates

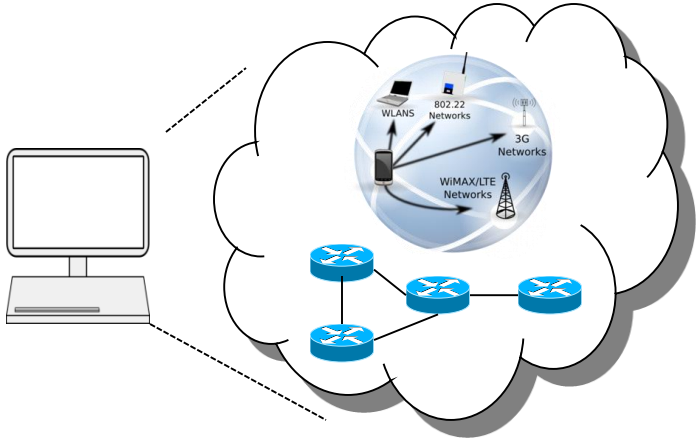
- Project provides three annual software releases
- Users interact on mailing lists and using Bugzilla bug tracker
- Code may be proposed for merge
  - Code reviews occur on a Google site
- Maintainers (one for each module) fix or delegate bugs, participate in reviews
- Project has been conducting annual workshop and developer meeting around SIMUTools through 2013
  - ns-3 Annual Meeting in Atlanta, May 2014
- Google Summer of Code (March-August) five of the past six summers

ns-3: An Open Source Network Simulator

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



model developers



ns-3 software

Research

NS-3 Consortium

ns-3 maintainers

Education

Goals of the NS-3 Consortium

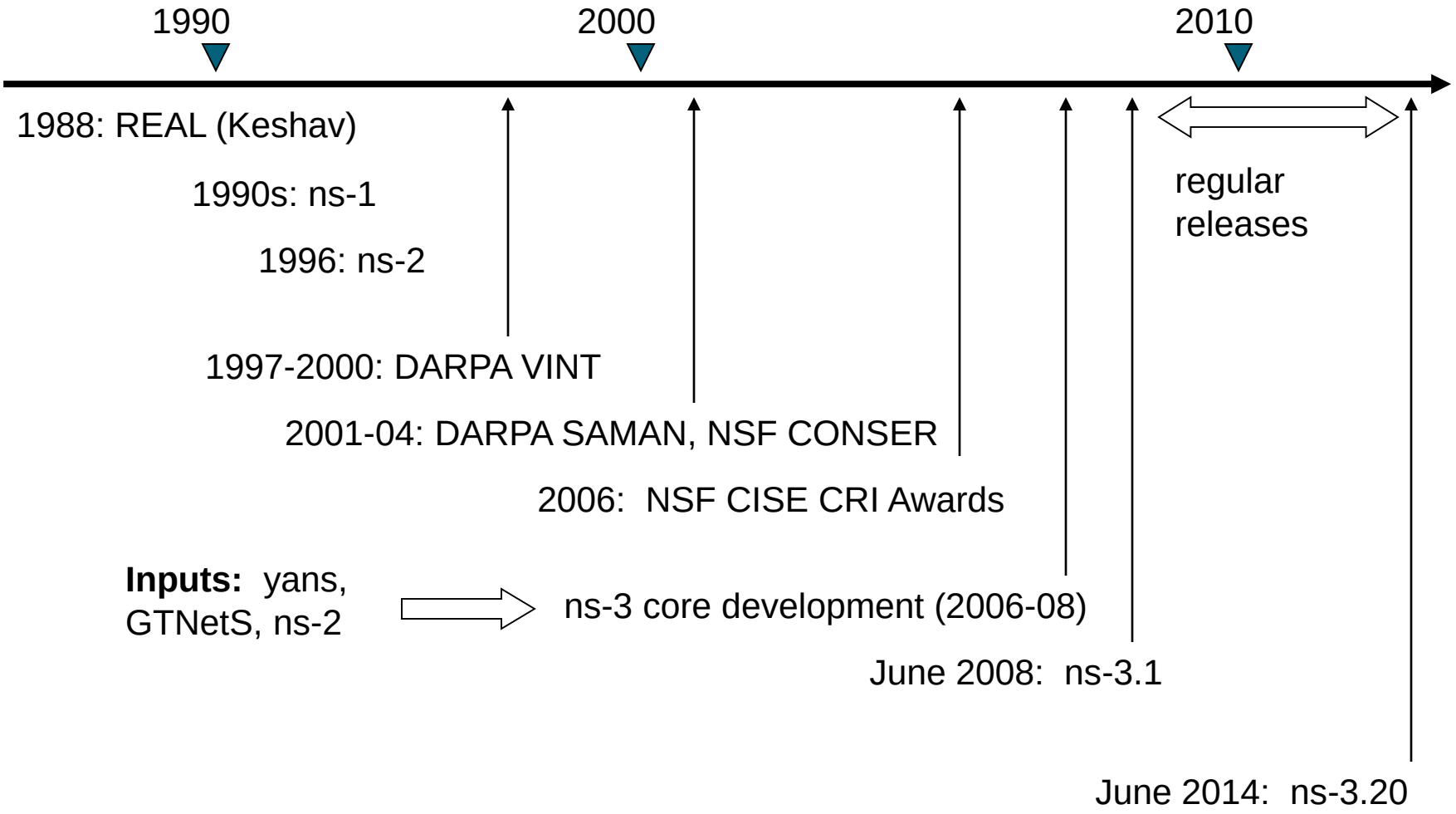
- The NS-3 Consortium is a collection of organizations cooperating to support and develop the ns-3 software.
- It operates in support of the open source project
  - by providing a point of contact between industrial members and ns-3 developers,
  - by sponsoring events in support of ns-3 such as users' 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.

Acknowledgment of support

Agenda

- ns-3 project overview
  - What is ns-3?
  - Why use ns-3?
  - Project organization
  - Relationship to ns-2
  - Future directions
- Getting started with ns-3

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 2014: ns-3.20

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

Agenda

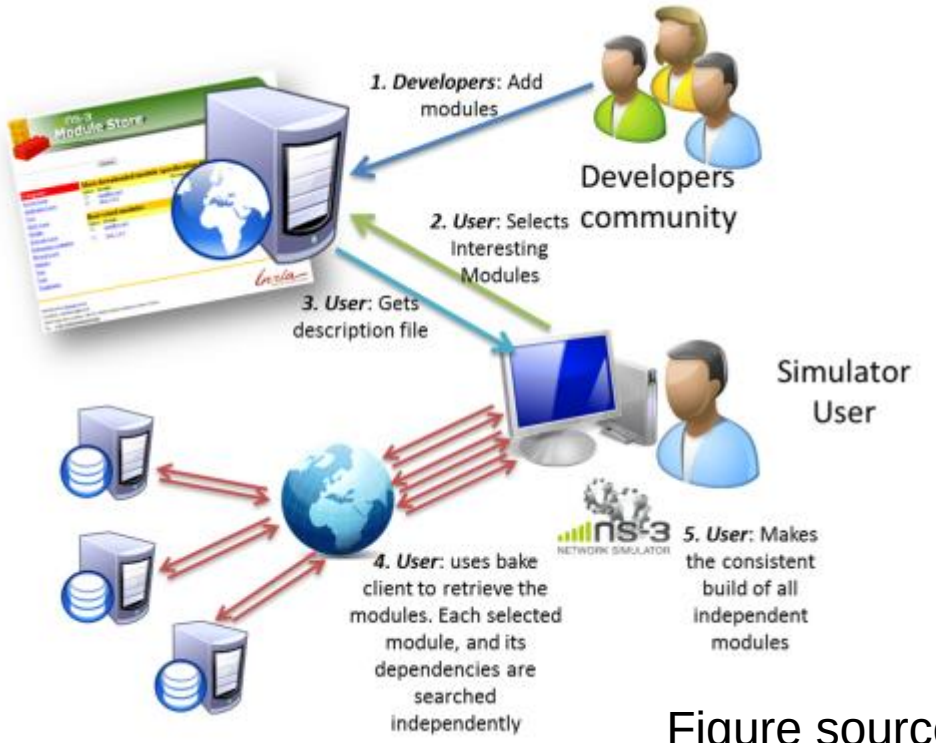
- ns-3 project overview
  - What is ns-3?
  - Why use ns-3?
  - Project organization
  - Relationship to ns-2
  - Future directions
- Getting started with ns-3

Development Priorities

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

Modularity

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



"bake" tool (Lacage and Camara)

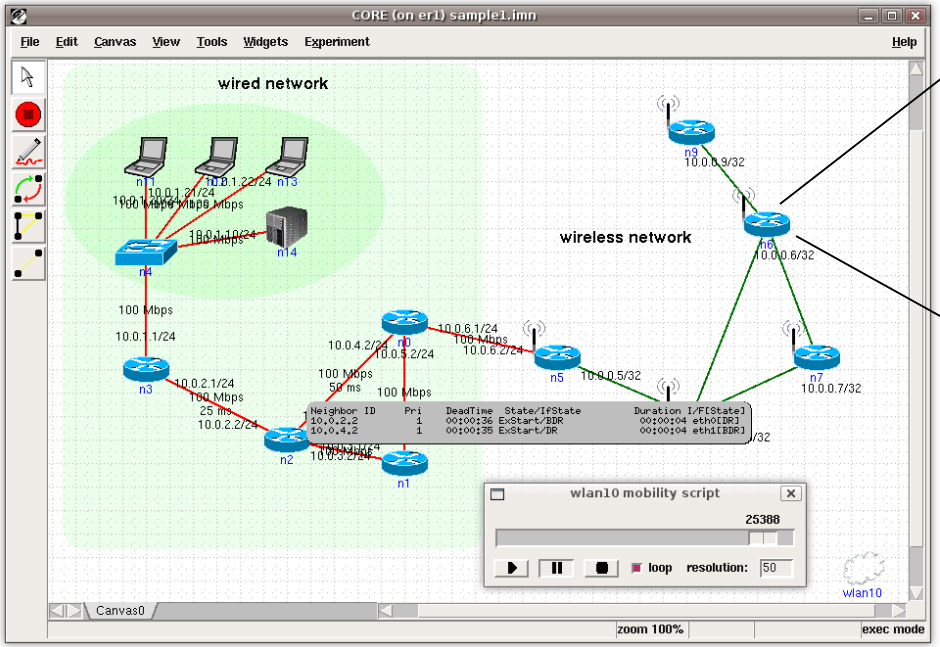
Components:

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

Figure source: Daniel Camara

Container-based Integration

- Common Open Research Emulator (CORE)
  - http://pf.itd.nrl.navy.mil
- Python-based framework using ns-3 Python bindings, distributed computing library, and ns-3 TapBridge framework



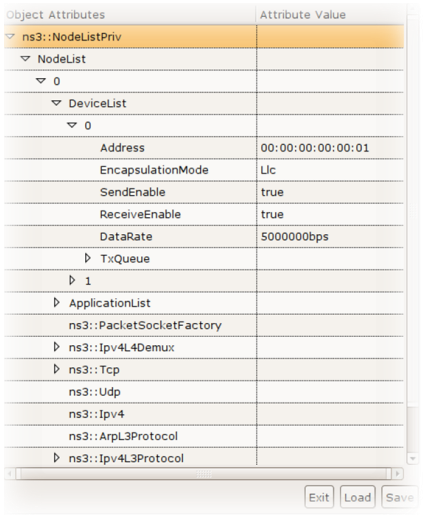


Figure source:
Jeff Ahrenholz

General issues with hybrid environments

- Ease of use
  - Configuration management and coherence
  - Information coordination (two sets of state)
    - e.g. IP/MAC address coordination
  - Output data exists in two domains
  - Debugging
- Error-free operation (avoidance of misuse)
  - Synchronization, information sharing, exception handling
    - Checkpoints for execution bring-up
    - Inoperative commands within an execution domain
    - Deal with run-time errors
  - Soft performance degradation (CPU) and time discontinuities

Network Experiment Management Framework (NEPI)

- Network experiment management framework to automate experiment life-cycle
- Allows scenarios involving heterogeneous resources (ns-3, PlanetLab, netns, …)
- Wiki: http://nepi.inria.fr

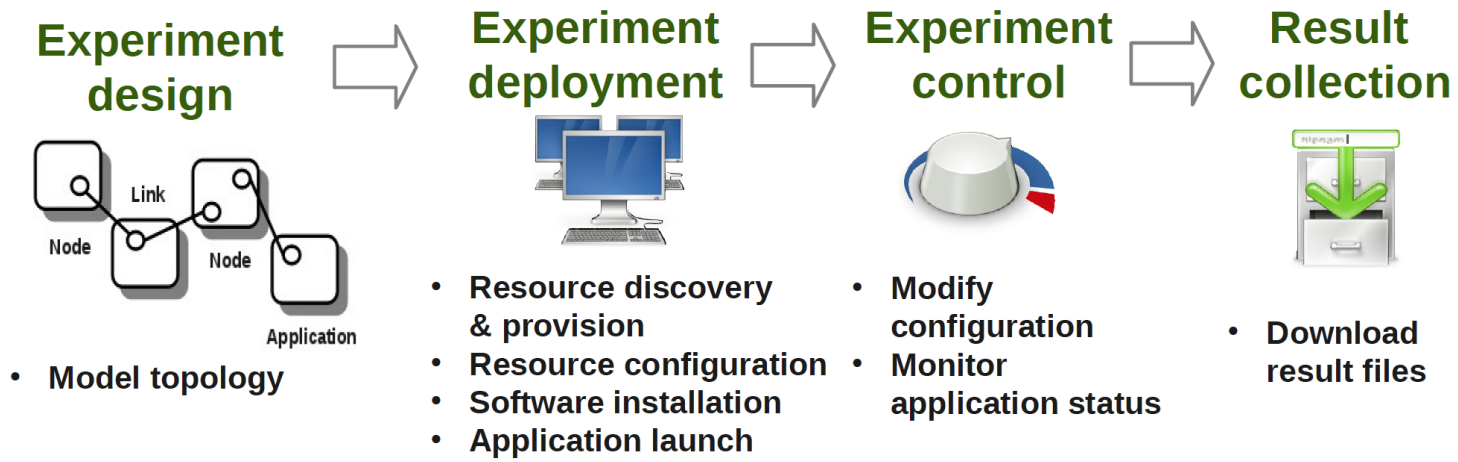


Figure source: Alina Quereilhac, INRIA

SAFE: Simulation Automation Framework

- Data collection, transient analysis, management of independent replications, graphical configuration and visualization
- In ns-2 realm, similar to projects like ANSWER, ns2measure, and Akaroa2

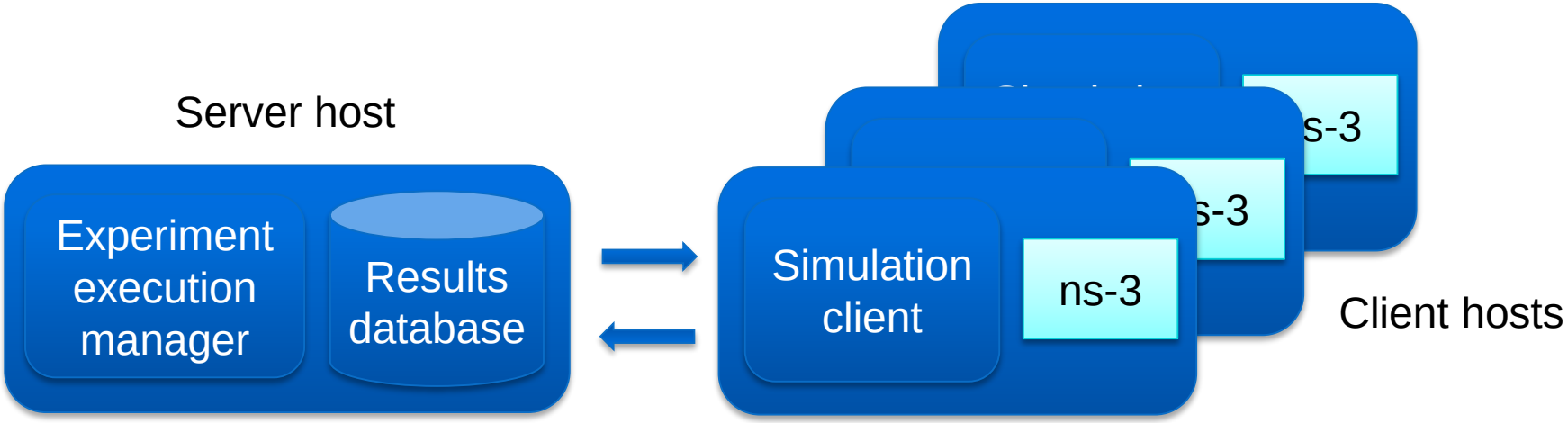
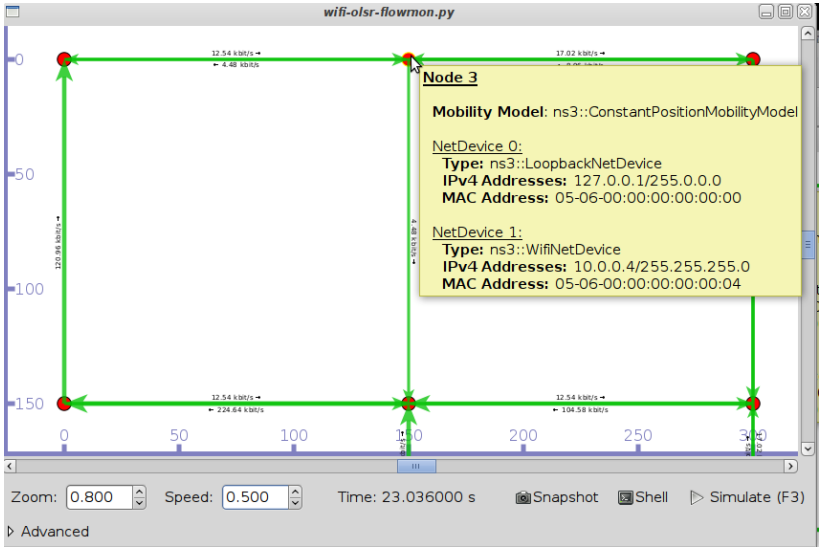


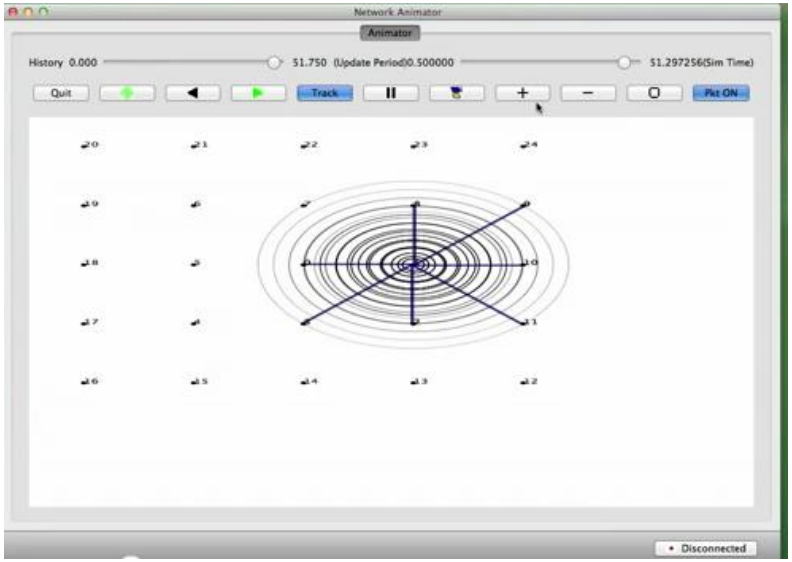
Figure source: Felipe Perrone

Usability

- Animation and visualization



PyVis (Carneiro)



NetAnim (Riley and Abraham)

- Linkage to external tools (topology, mobility, statistics)
- Improved helper APIs

Agenda

- ns-3 project overview
  - What is ns-3?
  - Why use ns-3?
  - Project organization
  - Relationship to ns-2
  - Future directions
- Getting started with ns-3

Getting started with ns-3

- Finding what you need
- Contributing to the project

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

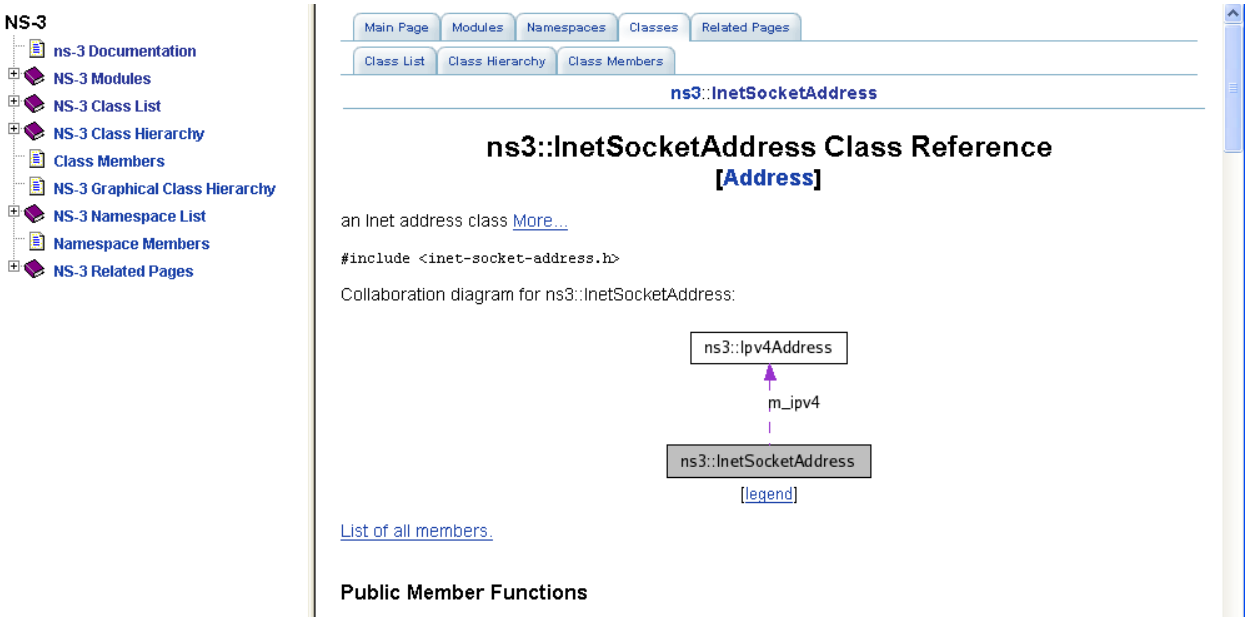
IRC: #ns-3 at freenode.net

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 (March 2013)
    - http://www.nsnam.org/consortium/activities/annual-meeting-march-2013/
- 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
  - http://www.stack.nl/~dimitri/doxygen/



Reading existing code

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

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)?
  - No

Contributing

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

New project ideas

- Visit the wiki under "Project Ideas" tab
  - http://www.nsnam.org/wiki/Project_Ideas
- Students, consider to apply for Google Summer of Code 2015
  - A 10-week summer job that mentors a student project on ns-3
  - Students apply in March 2015 timeframe
  - http://www.nsnam.org/wiki/GSOC2014Projects

Questions?