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

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

- **applications**
  - internet (IPv4/v6)

- **protocols**
  - aodv
  - dsdv
  - olsr
  - mpi
  - mobility
  - propagation
  - network
  - core
  - energy

- **utilities**
  - visualizer
  - config-store
  - flow-monitor
  - netanim
  - stats
  - topology-read
  - nix-vector-routing
  - openflow
  - BRITE

NS-3 Introduction
July 2014
Current models

- Bridge
- CSMA
- EMU
- Point-to-point

Devices:
- Bridge
- Mac
- Tuni
- Network

Applications:
- Internet (IPv4/v6)

Protocols:
- AODV
- DSVD
- OLSR

Energy:
- Packets
- Packet Tags
- Packet Headers
- Pcap/ascii file writing

Utilities:
- Click
- Stats
- Topology-read
- BRITE

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

Ir-wpan
Wifi
Wimax
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

**NS-3 Introduction**  
**July 2014**
Examples of recent publications


NS-3 Introduction
July 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 Miletic, Spring 2013, also Spring 2012
  - RM-RiTeh, Dr. Mladen Tomić and Vedran Miletic, 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*
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

- 1988: REAL (Keshav)
- 1990s: ns-1
  - 1996: ns-2
- 1997-2000: DARPA VINT
- 2001-04: DARPA SAMAN, NSF CONSER
- 2006: NSF CISE CRI Awards
- Inputs: yans, GTNetS, ns-2
- ns-3 core development (2006-08)
  - June 2008: ns-3.1
- June 2014: ns-3.20

Regular releases
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)

• Linkage to external tools (topology, mobility, statistics)

NetAnim (Riley and Abraham)

• 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

• Does ns-3 support Eclipse or other IDEs?
  – Instructions have been contributed by users

• 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
Questions?