





NS-3 Tutorial

Tom Henderson (University of Washington and Boeing Research & Technology) Mathieu Lacage (Alcméon)

March 2013



ns-3 tutorial agenda

• 13h00-15h00: Getting started with ns-3

- Overview of software and models
- Basic structure of the core and important models
- Running and understanding an existing example
- Animation and visualization
- 15h00-15h30: 30-minute coffee break

• 15h30-17h00: Going further with ns-3

- Writing and debugging your own examples
- Integrating other tools and libraries
- Parallel simulations
- Emulation, virtual machine and testbed integration
- Getting help and getting involved



Preliminaries

- ns-3 is written in C++, with bindings available for Python
 - simulation programs are C++ executables or Python programs
 - ~300,000 lines of mostly 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
- ns-3 is not backwards-compatible with ns-2



Preliminaries (cont.)

- Where do I get ns-3? -http://www.nsnam.org
- Where do I get today's code?
 http://www.nsnam.org/release/ns-allinone-3.16.tar.bz2



What have people done with ns-3?

- ~300 publications to date
 - search of 'ns-3 simulator' on IEEE and ACM digital libraries

		1		
1814	IEEE/ACM TRANSACTIONS ON NETWORKINO, VOL. 20, NO. 6, DECEMBER 2012		Wireless Netw (2011) 17:1775-1794 DOI 10.1007/s11276-011-0377-0	
FSR: Formal Analysis an	d Implementation Toolkit			
for Safe Interd	omain Routing	1		
for Safe Interdomain Routing		1	Message delivery in heterogeneous	networks prone to episodic
Anduo Wang, Limin Jia, Member, IEEE, Wenchao Zhou, Yiqing Ren, Boon Thau Loo,		1	connectivity	networks prone to episodie
Jennifer Rexford, Senior Member, IEEE, Vive	k Nigam, Andre Scedrov, and Carolyn Talcott	1	connectivity	
Abtract—Interdomain routing stitches the disparate parts of the Internet together, making protocol stability a critical issue to both researchers and practitioners. We researchers create safety proofs and counterexamples by hand and build simulators and prototypes to explore protocol dynamics. Similarly, setwork operators analyze their router configurations inamanality or using	Policy Configurations		Rao Naveed Bin Rais - Thierry Turletti - Katia Obraczka	
homegrown tools. In this paper, we present a comprehensive toollif 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 func-	Event Implementation Analysis Result		Published online: 17 August 2011 © Springer Science+Business Media, LLC 2011	
tions from the same algebraic representation of routing policy. We show that routing algebra has a start of random to both having constraints (to perform a side) with SMT (Joherri and Our effective captions with the start of	safety [4], [3]-[11], [33]. While our understanding of BGP safety has improved dramatically in the past decade, each re- search study still proceeds independently		Abstract We present an efficient message delivery framework, called McDella, which enables communication in an intermet connecting hereorgeneous networks that is prone to disruptions in connectivity. McDella is comple- mentary to the IRTP's Bundle Architecture: besides its ability to store messages for unavailable destinations, McDella can bridge the connectivity app between infra- medication and the store intervention of the store of the stor	McDeHa's ability to operate in environmen a diverse set of interconnected networks a performance through extensive simulations of scenarios with realistic synthetic and re ces. Our results show significant improver delivery ratio and a significant decrease in a delay in the face of episodic connectivity.
Index Terms-Communications technologoy, declarative net- working, formal analysis, routing algebra.	To aid the design, analysis, and evaluation of safe interdo-	1	structure-based and multi-hop infrastructure-less networks. It benefits from network heterogeneity (e.g., nodes sup-	onstrate that MeDeHa supports different le of-service through traffic differentiation
	main routing, we propose the Formally Safe Routing (FSR) toolkit. FSR serves two important communities. For re-	1	porting more than one network and nodes having diverse	prioritization.
I INTRODUCTION	searchers, FSR automates important parts of the design process	1	resources) to improve message delivery. For example, in	
	and provides a common framework for describing, evaluating,	1	IEEE 802.11 networks, participating nodes may use both	Keywords Disruption tolerance - Episodi
T HE INTERNET'S global routing system does not neces-	and comparing new safety guidelines. For network opera-	1	infrastructure- and ad-hoc modes to deliver data to other-	Heterogeneous networks · Node relaying · S
sarily converge, depending on how the Border Gateway Protocol (BGP) policies of individual networks are config-	tors, FSR automates the analysis of internal router (iBGP) and border gateway (eBGP) configurations for safety viola-	1	wise unavailable destinations. It also employs opportunistic	forward - DTN routing
ured. Since protocol oscillations cause serious performance	tions. For both communities, FSR automatically generates	1	routing to support nodes with episodic connectivity. One of MeDeHa's key features is that any MeDeHa node can relay	
disruptions and router overhead, researchers devote signifi-	realistic protocol implementations to evaluate real network	1	data to any destination and can act as a gateway to make	1 Introduction
cant attention to BGP stability (or "safety"). Abstract formal	configurations (e.g., to study convergence time) prior to actual deployment. The ideas underlying FSR also unify research in	1	two networks inter-operate or to connect to the backbone	
models of BGP [12]-[15], [36] allow researchers to explore how local policies affect BGP stability and identify policy	routing algebras [13], [36] with recent advances in declarative	1	network. The network is able to store data destined to	It is envisioned that the Internet of the futur
guidelines that, if universally adopted by ISPs, ensure global		1	temporarily unavailable nodes till the time of their expiry.	heterogeneous not only due to the wide
	of safe interdomain routing.	1	This time period depends upon current storage availability as well as quality-of-service needs (e.g., delivery delay	 devices it interconnects, but also in terms of networks it comprises. Figure 1 illustrates
Manuscript received May 23, 2011; accepted January 21, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. M. Mao, Date of pub-	Given policy configurations as input, FSR produces an analysis of safety properties and a distributed protocol imple-	1	bounds) imposed by the application. We showcase	range from wired- and wireless backbones (
lication March 14, 2012; date of current version December 13, 2012. This work	mentation, as shown in Fig. 1. FSR has three main underlying	1		wireless mesh networks) to wireless infra
was supported in part by the NSF under Grants CCF-0820208, CNS-0830949, CNS-0845552, CNS-1040672, CPS-0932397, IIS-0812270, and TC-0905607;	technologies.	1		and ad-hoc networks (e.g., MANETs). On
the AFOSR under Grants FA9350-08-1-0332 and FA9350-09-1-0643; the ONR under Grants N00014-09-1-0770 and N00014-11-1-0555; a gift from	 Policy configuration as algebra: Our extensions to routing algebra [13], [36] allow researchers and network operators 	1	R. N. B. Rais (54)	current and emerging applications, such
Cisco System; and the A. von Humboldt Foundation.	to express policy configurations in an abstract alge-	1	COMSATS Institute of Information Technology (CIIT), Labore, Pakistan	response, environmental monitoring, smar
A. Wang, W. Zhou, Y. Ren, B. T. Loo, and A. Scedrov are with the University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail:	braic form. These configurations can be anything from	1	e-mail: naveedbinrais@ciitlahore.edu.pk	 (e.g., smart offices, homes, museums, etc.) networks, among others imply frequent
andun@soas.upenn.edu; wenchaon@soas.upenn.edu; yiqingr@soas.upenn.edu;	high-level policy guidelines (e.g., proposed constraints	1	T. Turkti	long-lived disruptions in connectivity. The
boonloo@seas.uperm.edu; scedrov@math.uperm.edu). L. Jia is with Camerie Mellon University. Pittsburgh. PA 15213 USA (e-mail: 10.1000/0000000000000000000000000000000	that a researcher wants to study) or a completely specified policy instance [e.g., an iBGP configuration or a multi-au-	1	1. Turiem INRIA, Sonhia Antipolis, Prance	runtion- or delay-tolerant networks (DTI
liminia@cmt.edu). J. Reaford is with Princeton University, Princeton, NJ 08540 USA (e-mail:	tonomous-system (AS) network that an operator wants to	1	e-mail: thiery.tufetti@sophia.inria.fr	become an important component of future
irest (i cs princeton eds).	analyze]. Router configuration files can be automatically	1	K. Obraczka	Seamless interoperability among hete
V. Nigam is with the Computer Science Department, Ludwig-Maximilians University of Munich, Munich 80539, Germany.	translated into the algebraic representation, easing the	1	University of California, Santa Cruz, CA, USA	works is a challenging problem as these
C. Talcott is with the SRI International, Menio Park, CA 94025 USA (e-mail:	 adoption of FSR. Safety analysis: To automatically analyze the policy 	1	e-mail: kati a@ soe.ucsc.edu	have very different characteristics. Node div
clt@csl.uri.com). Digital Object Identifier 10.1109/TNET 2012.2187924	 sapely analysis: 10 automatically analyze the policy configuration, FSR reduces the convergence proof to a 	1		

1063-6692/\$31.00 C 2012 IEEE

resent an efficient message delivery MeDeHa's ability to operate in environments consisting of MeDeHa, which enables communication a diverse set of interconnected networks and evaluate its a diverse set of interconnected networks and evaluate its nnecting heterogeneous networks that is performance through extensive simulations using a variety ns in connectivity. MeDeHa is compleof scenarios with realistic synthetic and real mobility tra or scenarios with realistic synthetic and real mobility tra-ces. Our results show significant improvement in average delivery ratio and a significant decrease in average delivery IRTF's Bundle Architecture: besides its messages for unavailable destinations, idge the connectivity gap between infra- delay in the face of episodic connectivity. We also demonstrate that MeDeHa supports different levels of quality-of-service through traffic differentiation and message re-less networks network heterogeneity (e.g., nodes supn one network and nodes having diverse prioritization. prove message delivery. For example, in

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

1 Introduction

use-operate of to connect to the backstone levels, is a blue to store data destined to a list envisioned that the Internet of the future will be highly walable nodes till the time of their expiry. heterogeneous not only due to the wide variety of end of depends upon current storage availability devices it interconnects, but also in terms of the underlying heterogeneous not only due to the wide variety of end 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 disruntion- or delay-tolerant networks (DTNs) will likely become an important component of future internetworks Seamless interoperability among heterogeneous net-works is a challenging problem as these networks may have very different characteristics. Node diversity may also

Springer

Augmenting Data Center Networks with Multi-Gigabit Wireless Links

Daniel Halperin+, Srikanth Kandula+, Jitendra Padhye+, Paramvir Bahl+, and David Wetherall-Microsoft Research+ and University of Washington+

hstract - The 60 GHz wireless technology that is now emerging A unit were into locate dimensional and the second which takes can be concentrally at multi-Gips mass on type-fract (Gi) which can be wide DC stretchs can be used to show the definition of the simulation fraction fraction fraction for the simulation of the simulation fraction fraction fraction for the simulation of the simulation

Categories and Subject Descriptors C.2.1 [Computer-Communication Networks]: Network Archi-tecture and Design-Wireless Communication

General Terms

Design Experimentation Measurement Performance

1. INTRODUCTION

Millimeter wavelength wireless technology is rapidly being de-veloped. Spectrum between 37-64 (IHz, colloquially known as the 60 GHz hand, is available world-wide for unificensed use. The hand contains over 80 times the bandwidth available for 802.11b/g at 2.4 GHz, and supports devices with multi. Gbns data rates. Fur densely, because the signal attenuates rapidly due to the high fredensely, because the signal attenuates rapidly due to the high tre-quency. The VLB technology has now matured to the point where 60 GHz radio hardware can be built using CMOS technology, and companies like Silbeam [26] promise to deliver 60 GHz devices at its sith as 10 pro-unit at OEM quantities. In summary, 60 GHz tech-nology can lead to dense, high-bandwidth wireless connectivity at low cost

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without for provided that copies are not made or distributed for profits or commercial abattage and that copies bear thin notice and the full citation on the finit page. To equite prior opecific mphilith, to provide on servers or to redistribute to lists, rouguite prior specific specific and servers or to medistribute to lists. To equip the prior specific specific specific servers and the start that the specific specific specific specific servers or to medistribute to lists. The specific specific specific servers or to medistribute to lists. The specific servers of the specifi and/or a fee. permission and/or a tee. SIGCOMM*11, August 15–19, 2011, Toronto, Ontario, Canada. Conversible 2011 ACM 978, 1.4503.0707.0111/08...510.00

to used, but the terminory matching matching the terminor of model pairs to point links. A common scenario is home exterimation, e.g., a Bla-Ray player that communicates which says with a nearby televitation instand of using hubby HDML cables. In this paper, we consider the novel possibility of using GO GHe links in a data center (DC), to asymmet the wheed network. This is a promising approach to explore for several reasons. First, we note that the matchines is a DC are densely player. In which we have the several termination of the several reasons. First, we not that the maximum is a DC are shownly packet, so writely down that provide blackwidth over the transparse an abu-diant that provide blackwidth over the transparse and abu-ph and explorement row around information (abu-ti that abuption). The Hinss of abuption (abuption) ableses by mounding dO GD is rules on using of racks. Findship with the downly of the DD abuption (abuption) and the problem such as ablighting directional such as all reactions, and provide the downly of the downly of the downly of the problem such as a simplifying directional such as all reactions, and problem such as a balanced abuption of the downly of the problem such as a simplifying directional such as all reactions, and problem such as a simplifying direction and the association of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of the problem such as a simplifying direction of the downly of t

To date, 60 GHz technology has been explored for isolated poin

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 otspot that hinders some DC application. Recent research tack hotipot that hinders some IX: application. Necent research tack-les this problem by combining many more links and winches will variants of multipath routing so that the core of the network is no longer oversubscribed [11, 8, 9]. Of course, this benefit corens will large material cost and implementation complexity [15]. Some dc-signs require 'somary wires that cabling becomes a challenge [1], and most require 'fork lift' [3] apprades to the entire infrastructure. In prior work [15], we argued instead for a more modest addi on of links to relieve hotspots and boost application performance

tion of thicks to train whether biophysical mater competitudies performance. The blacks, called fory exception by the base networks to allow its basepoint. When the mattime matrix is space its cardy a structure of the structure cardy structure performance, which are the card of basiling a radiu train of the structure of the str set up flyways between ToR switches that provide added band width as needed. Other researchers have explored use of fiber optic cables and

Other is ideations table explored use of nitree optic clanks and BMSS switches [7, 36] for exampling hyperay. We helice that 60 GHz flyways are an attractive choice because winters devices simplify the identical system or winter changes are needed. Furthermore, 60 GHz technology is filely to become insegnesive as it is commolitated by consumer applications, while optical witches as no cold writes to de-vices can introduce additional issues as well—for example, with dynamic lopology, the network management may become more

NS-3 Consortium Meeting March 2013



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 A, and EECS 983 & Dr. James Sterbenz, 2010 2012
- Aalto University Dose 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.



Software introduction

- Download the latest release
 - wget http://www.nsnam.org/releases/ns-allinone-3.16.tar.bz2
 - tar xjf ns-allinone-3.16.tar.bz2
- Clone the latest development code

- hg clone http://code.nsnam.org/ns-3-allinone

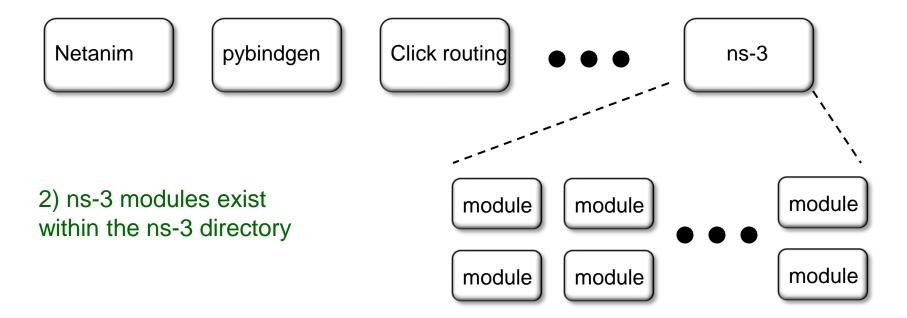
Q. What is "hg clone"?

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

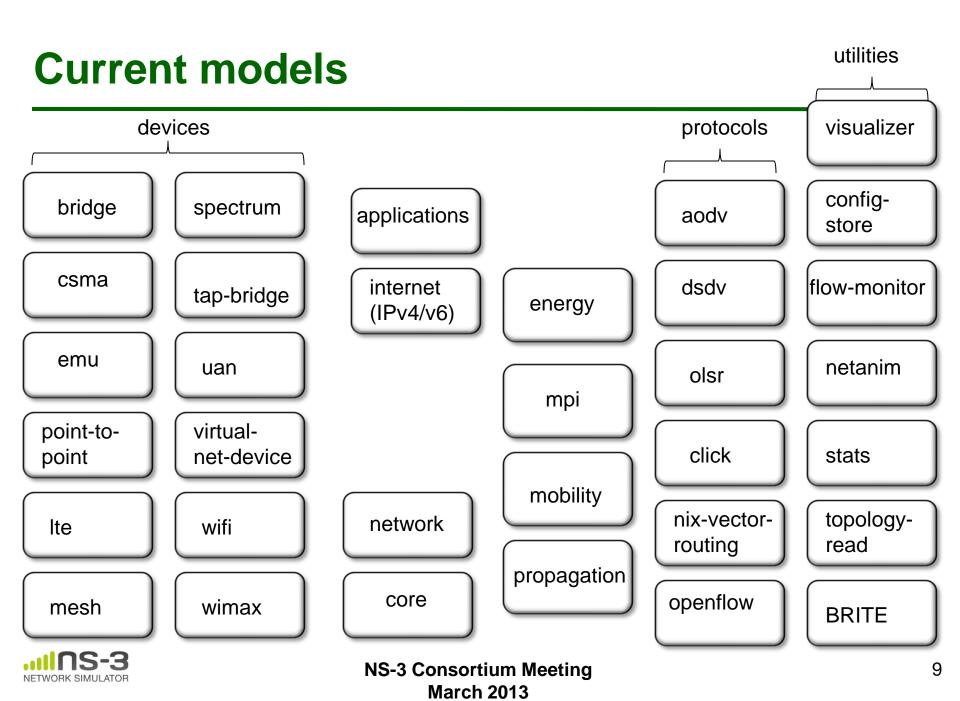


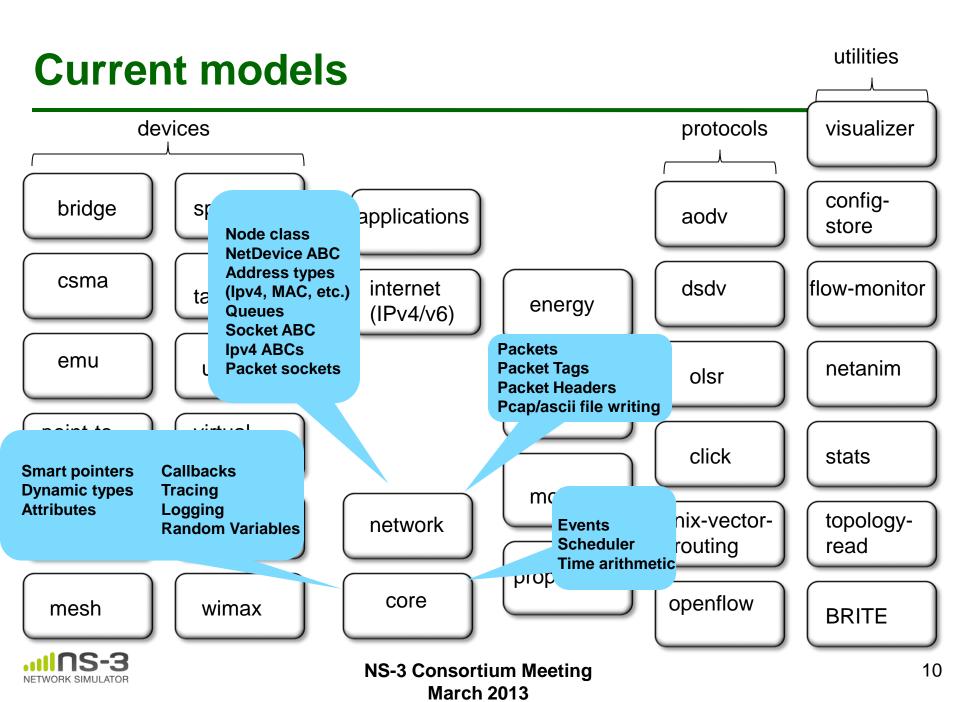
Software organization

- Two levels of ns-3 software and libraries
 - 1) Several supporting libraries, not system-installed, can be in parallel to ns-3









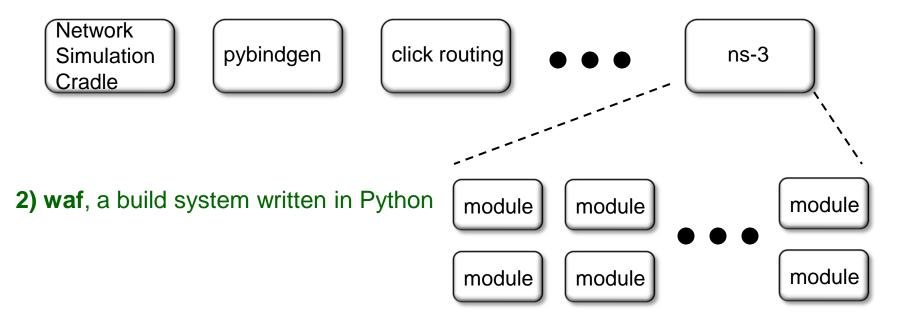
Module organization

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



Software building

- Two levels of ns-3 build
- 1) build.py (a custom Python build script to control an ordered build of ns-3 and its libraries)





ns-3 uses the 'waf' build system

- Waf is a Python-based framework for configuring, compiling and installing applications.
 - It is a replacement for other tools such as Autotools, Scons, CMake or Ant
 - -http://code.google.com/p/waf/
- For those familiar with autotools:
- configure --> ./waf configure
- make \longrightarrow ./waf build



waf configuration

- Key waf configuration examples
 - ./waf configure
 - --enable-examples
 - --enable-tests
 - --disable-python
 - --enable-modules
- Whenever build scripts change, need to reconfigure

```
Demo: ./waf --help
   ./waf configure --enable-examples --
enable-tests --enable-modules='core'
Look at: build/c4che/_cache.py
```



wscript example

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-
def build(bld):
    obj = bld.create_ns3_module('csma', ['network', 'applications'])
    obj.source = [
        'model/backoff.cc',
        'model/csma-net-device.cc',
        'model/csma-channel.cc',
        'helper/csma-helper.cc',
   headers = bld.new task gen(features=['ns3header'])
   headers.module = 'csma'
   headers.source = [
        'model/backoff.h',
        'model/csma-net-device.h',
        'model/csma-channel.h',
        'helper/csma-helper.h',
        ]
    if bld.env['ENABLE EXAMPLES']:
       bld.add_subdirs('examples')
```

```
bld.ns3_python_bindings()
```



waf build

- Once project is configured, can build via ./waf build Or ./waf
- waf will build in parallel on multiple cores
- waf displays modules built at end of build

Demo: ./waf build

Look at: build/ libraries and executables



NS-3 Consortium Meeting March 2013

Running programs

- ./waf shell provides a special shell for running programs
 - -Sets key environment variables
 - ./waf --run sample-simulator
 - ./waf --pyrun src/core/examples/samplesimulator.py



NS-3 Consortium Meeting March 2013

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() gets it all started
- Simulation stops at specific time or when events end



Simulator example

```
#include <iostream>
#include "ns3/simulator.h"
#include "ns3/nstime.h"
#include "ns3/command-line.h"
#include "ns3/double.h"
#include "ns3/random-variable-stream.h"
```

```
using namespace ns3;
```

```
int main (int argc, char *argv[])
 CommandLine cmd;
 cmd.Parse (argc, argv);
 MyModel model;
 Ptr<UniformRandomVariable> v = CreateObject<UniformRandomVariable> ();
 v->SetAttribute ("Min", DoubleValue (10));
 v->SetAttribute ("Max", DoubleValue (20));
 Simulator::Schedule (Seconds (10.0), &ExampleFunction, &model);
 Simulator::Schedule (Seconds (v->GetValue ()), &RandomFunction);
 EventId id = Simulator::Schedule (Seconds (30.0), &CancelledEvent);
 Simulator::Cancel (id);
 Simulator::Run ();
 Simulator::Destroy ();
```



Simulator example (in Python)

```
# Python version of sample-simulator.cc
```

import ns.core

```
def main(dummy argv):
   model = MyModel()
   v = ns.core.UniformRandomVariable()
   v.SetAttribute("Min", ns.core.DoubleValue (10))
   v.SetAttribute("Max", ns.core.DoubleValue (20))
   ns.core.Simulator.Schedule(ns.core.Seconds(10.0), ExampleFunction, model)
   ns.core.Simulator.Schedule(ns.core.Seconds(v.GetValue()), RandomFunction, model)
   id = ns.core.Simulator.Schedule(ns.core.Seconds(30.0), CancelledEvent)
   ns.core.Simulator.Cancel(id)
   ns.core.Simulator.Run()
   ns.core.Simulator.Destroy()
if name == ' main ':
   import sys
   main(sys.argv)
```



Command-line arguments

 Add CommandLine to your program if you want command-line argument parsing

```
int main (int argc, char *argv[])
{
   CommandLine cmd;
   cmd.Parse (argc, argv);
```

 Passing --PrintHelp to programs will display command line options, if CommandLine is enabled

./waf --run "sample-simulator --PrintHelp"

--PrintHelp: Print this help message. --PrintGroups: Print the list of groups. --PrintTypeIds: Print all TypeIds. --PrintGroup=[group]: Print all TypeIds of group. --PrintAttributes=[typeid]: Print all attributes of typeid. --PrintGlobals: Print the list of globals.



NS-3 Consortium Meeting March 2013

Time in ns-3

- Time is stored as a large integer in ns-3

 Avoid floating point discrepancies across platforms
- Special Time classes are provided to manipulate time (such as standard operators)
- Default time resolution is nanoseconds, but can be set to other resolutions
- Time objects can be set by floating-point values and can export floating-point

double timeDouble = t.GetSeconds();



Events in ns-3

- Events are just function calls that execute at a simulated time
 - -i.e. callbacks
- Events have IDs to allow them to be cancelled or to test their status



Simulator and Schedulers

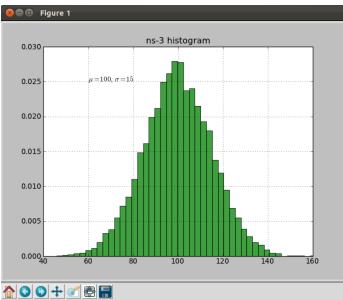
- The Simulator class holds a scheduler, and provides the API to schedule events, start, stop, and cleanup memory
- Several scheduler data structures (calendar, heap, list, map) are possible
- A "RealTime" simulation implementation is possible

-aligns the simulation time to wall-clock time



- Currently implemented distributions
 - Uniform: values uniformly distributed in an interval
 - Constant: value is always the same (not really random)
 - Sequential: return a sequential list of predefined values
 - Exponential: exponential distribution (poisson process)
 - Normal (gaussian), Log-Normal, Pareto, Weibull, triangular

```
# Demonstrate use of ns-3 as a random number generator integrated with
# 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()
```





NS-3 Consortium Meeting March 2013

Random variables and independent replications

 Many simulation uses involve running a number of *independent replications* of the same scenario

In ns-3, this is typically performed by incrementing the simulation *run number* – *not by changing seeds*



ns-3 random number generator

- Uses the MRG32k3a generator from Pierre L'Ecuyer
 - http://www.iro.umontreal.ca/~lecuyer/myftp/papers/str eams00.pdf
 - Period of PRNG is 3.1x10^57
- Partitions a pseudo-random number generator into <u>uncorrelated</u> streams and substreams
 - Each RandomVariableStream gets its own stream
 - This stream partitioned into substreams



Run number vs. seed

- If you increment the seed of the PRNG, the streams of random variable objects across different runs are not guaranteed to be uncorrelated
- If you fix the seed, but increment the run number, you will get an uncorrelated substream



Putting it together

• Example of scheduled event

```
int main (int argc, char *argv[])
{
   CommandLine cmd;
   cmd.Parse (argc, argv);
   MyModel model;
   Ptr<UniformRandomVariable> v = CreateObject<UniformRandomVariable> ();
   v->SetAttribute ("Min", DoubleValue (10));
   v->SetAttribute ("Max", DoubleValue (20));
   Simulator::Schedule (Seconds (10.0), &ExampleFunction, &model);
   Simulator::Schedule (Seconds (v->GetValue ()), &RandomFunction);
```

Demo real-time, command-line, random variables...



Build variations

- Configure a build type is done at waf configuration time
- debug build (default): all asserts and debugging code enabled

./waf -d debug configure

optimized

./waf -d optimized configure

static libraries

./waf --enable-static configure



Controlling the modular build

• One way to disable modules:

- ./waf configure --enable-modules='a','b','c'

- The .ns3rc file (found in utils/ directory) can be used to control the modules built
- Precedence in controlling build
 - 1) command line arguments
 - 2) .ns3rc in ns-3 top level directory
 - 3) .ns3rc in user's home directory

Demo how .ns3rc works



Building without wscript

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

Demo how programs can be built without wscripts



APIs

- Most of the ns-3 API is documented with Doxygen
 - -http://www.stack.nl/~dimitri/doxygen/





NS-3 Consortium Meeting March 2013

Review of topics covered

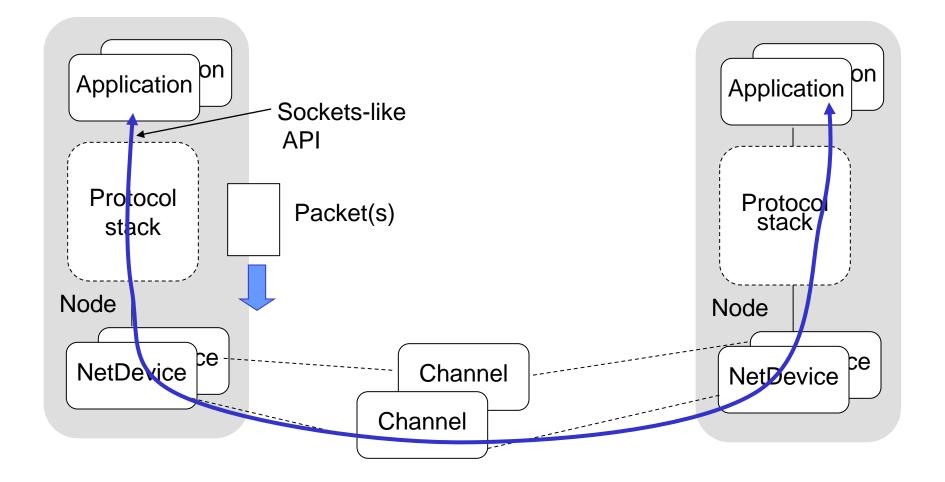
- Software layout
- Software build
- Library documentation
- Basic discrete-event simulation concepts
- Control of randomness
- Simulation time
- A simple C++ ns-3 program
- A simple Python ns-3 program



Walkthrough of WiFi Internet example



The basic model





Example program

- examples/wireless/wifi-simple-adhocgrid.cc
- examine wscript for necessary modules
 - 'internet', 'mobility', 'wifi', 'config-store',
 'tools'
 - we'll add 'visualizer'

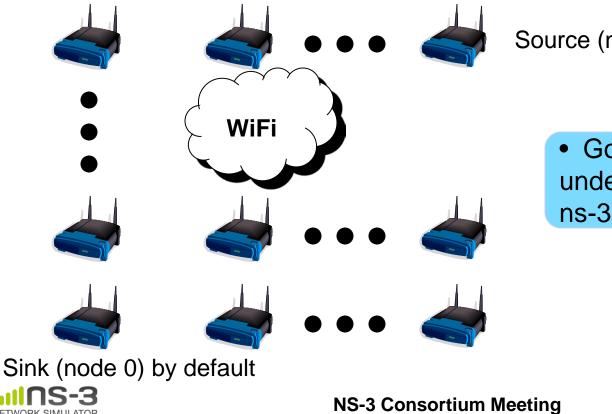
• ./waf configure --enable-examples -enable-modules=...



Example program

- (5x5) grid of WiFi ad hoc nodes
- OLSR packet routing
- Try to send packet from one node to another

March 2013



Source (node 24) by default

 Goal is to read and understand the high-level ns-3 API

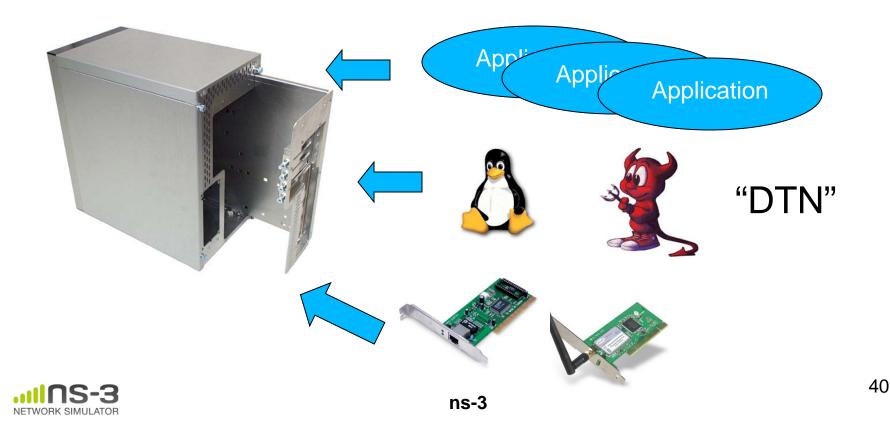
Key objects in the simulator are Nodes, Packets, and Channels

Nodes contain Applications, "stacks", and NetDevices



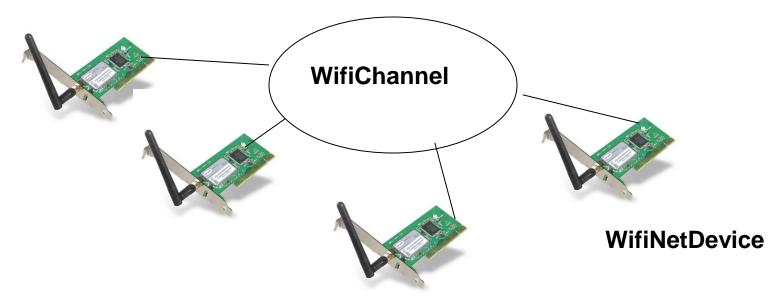


A Node is a shell of a computer to which applications, stacks, and NICs are added



NetDevices and Channels

NetDevices are strongly bound to Channels of a matching type



Nodes are architected for multiple interfaces



Internet Stack

- Internet Stack
 - Provides IPv4 and some IPv6 models currently
- No non-IP stacks presently in ns-3
 - -but no dependency on IP in the devices, Node, Packet, etc.
 - -some activity on IEEE 802.15.4-based models



Other basic models in ns-3

- Devices
 - -WiFi, WiMAX, CSMA, Point-to-point, Bridge
- Error models and queues
- Applications
 - -echo servers, traffic generator
- Mobility models
- Packet routing
 - -OLSR, AODV, DSR, DSDV, Static, Nix-Vector, Global (link state)



ns-3 Packet

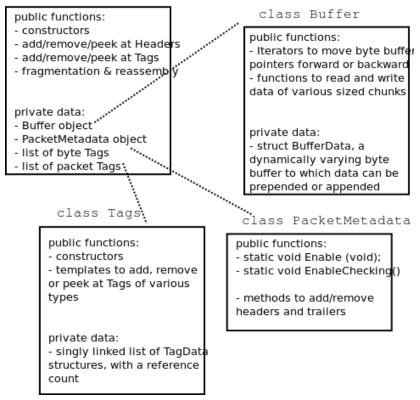
- Packet is an advanced data structure with the following capabilities
 - -Supports fragmentation and reassembly
 - -Supports real or virtual application data
 - -Extensible
 - -Serializable (for emulation)
 - -Supports pretty-printing
 - -Efficient (copy-on-write semantics)



ns-3 Packet structure

Analogous to an mbuf/skbuff

class Packet





Copy-on-write

Copy data bytes only as needed

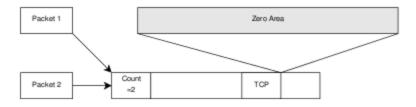


Figure 3.8: The TCP and the IP stacks hold references to a shared buffer.



Figure 3.9: The IP stack inserts the IP header, triggers an un-share operation, completes the insertion.



Structure of an ns-3 program

```
int main (int argc, char *argv[])
{
```

- // Set default attribute values
- // Parse command-line arguments
- // Configure the topology; nodes, channels, devices, mobility
- // Add (Internet) stack to nodes
- // Configure IP addressing and routing
- // Add and configure applications
- // Configure tracing
- // Run simulation



}

Review of example program

```
NodeContainer c;
c.Create (numNodes);
// The below set of helpers will help us to put together the wifi NICs we want
WifiHelper wifi;
if (verbose)
  {
    wifi.EnableLogComponents (); // Turn on all Wifi logging
  }
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
// set it to zero; otherwise, gain will be added
wifiPhy.Set ("RxGain", DoubleValue (-10) );
// ns-3 supports RadioTap and Prism tracing extensions for 802.11b
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT IEEE802 11 RADIO);
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());
// Add a non-QoS upper mac, and disable rate control
NgosWifiMacHelper wifiMac = NgosWifiMacHelper::Default ();
wifi.SetStandard (WIFI PHY STANDARD 80211b);
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode".StringValue (phyMode));
// Set it to adhoc mode
wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer devices = wifi.Install (wifiPhv. wifiMac. c):
MobilityHelper mobility;
```

NETWORK SIMULATOR

Helper API

- The ns-3 "helper API" provides a set of classes and methods that make common operations easier than using the low-level API
- Consists of:
 - container objects
 - helper classes
- The helper API is implemented using the lowlevel API
- Users are encouraged to contribute or propose improvements to the ns-3 helper API



Containers

- Containers are part of the ns-3 "helper API"
- Containers group similar objects, for convenience
 - They are often implemented using C++ std containers
- Container objects also are intended to provide more basic (typical) API



The Helper API (vs. low-level API)

- Is not generic
- Does not try to allow code reuse
- Provides simple 'syntactical sugar' to make simulation scripts look nicer and easier to read for network researchers
- Each function applies a single operation on a "set of same objects"



Helper Objects

- NodeContainer: vector of Ptr<Node>
- NetDeviceContainer: vector of Ptr<NetDevice>
- InternetStackHelper
- WifiHelper
- MobilityHelper
- OlsrHelper
- ... Each model provides a helper class



Example program

- (5x5) grid of WiFi ad hoc nodes
- OLSR packet routing
- Try to send packet from one node to another

Source (node 24) by default WiFi Let's look closely at how these objects are created Sink (node 0) by default 53 **NS-3 Consortium Meeting**

Installation onto containers

 Installing models into containers, and handling containers, is a key API theme

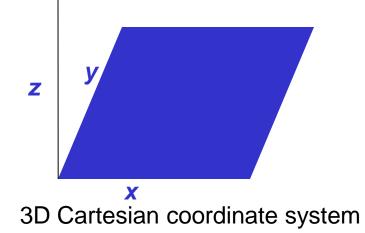
```
NodeContainer c;
c.Create (numNodes);
...
mobility.Install (c);
...
internet.Install (c);
```



. . .

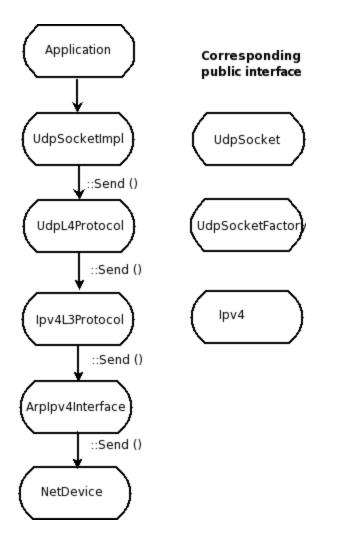
Mobility models in ns-3

- The MobilityModel interface:
 - void SetPosition (Vector pos)
 - Vector GetPosition ()
- StaticMobilityModel
 - Node is at a fixed location; does not move on its own
- RandomWaypointMobilityModel
 - (works inside a rectangular bounded area)
 - Node pauses for a certain random time
 - Node selects a random waypoint and speed
 - Node starts walking towards the waypoint
 - When waypoint is reached, goto first state
- RandomDirectionMobilityModel
 - works inside a rectangular bounded area)
 - Node selects a random direction and speed
 - Node walks in that direction until the edge
 - Node pauses for random time
 - Repeat





Internet stack



• The public interface of the Internet stack is defined (abstract base classes) in src/network/model directory

- The intent is to support multiple implementations
- The default ns-3 Internet stack is implemented in src/internet-stack





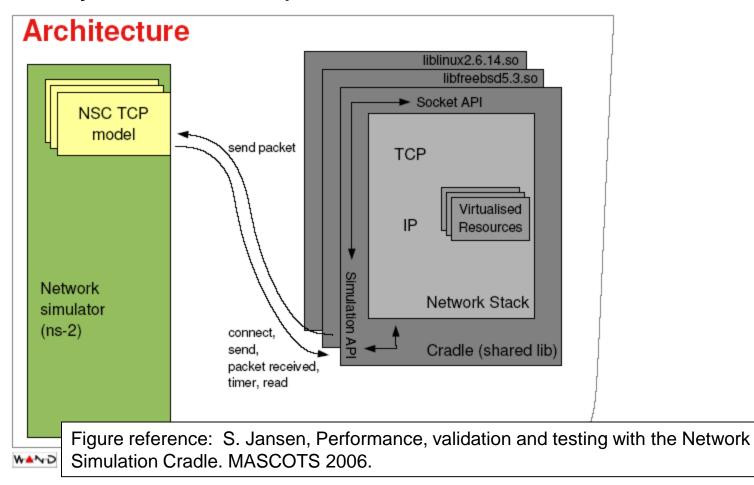
- Several options exist:
 - -native ns-3 TCP
 - Tahoe, Reno, NewReno (others in development)
 - -TCP simulation cradle (NSC)
 - -Use of virtual machines or DCE (more on this later)
- To enable NSC:

internetStack.SetNscStack ("liblinux2.6.26.so");



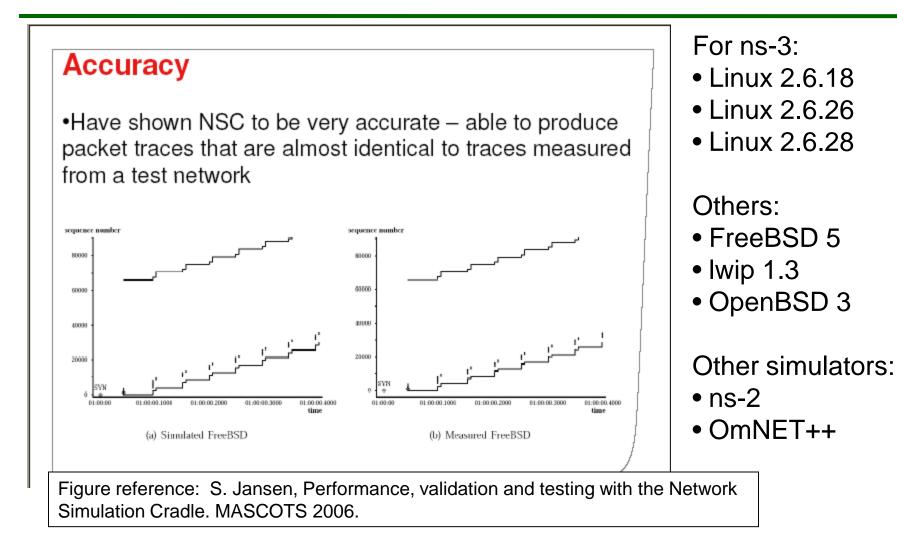
ns-3 simulation cradle

• Port by Florian Westphal of Sam Jansen's Ph.D. work





ns-3 simulation cradle





IPv4 address configuration

 An Ipv4 address helper can assign addresses to devices in a NetDevice container

```
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
csmaInterfaces = ipv4.Assign (csmaDevices);
```

•••

```
ipv4.NewNetwork (); // bumps network to 10.1.2.0
otherCsmaInterfaces = ipv4.Assign (otherCsmaDevices);
```



Applications and sockets

- In general, applications in ns-3 derive from the ns3::Application base class
 - A list of applications is stored in the ns3::NodeApplications are like processes
- Applications make use of a sockets-like API
 - –Application::Start () may call ns3::Socket::SendMsg() at a lower layer



Sockets API

```
Plain C sockets
                                            ns-3 sockets
                                            Ptr<Socket> sk =
int sk;
sk = socket(PF INET, SOCK DGRAM, 0);
                                            udpFactory->CreateSocket ();
struct sockaddr in src;
inet pton(AF INET,"0.0.0.0",&src.sin ad sk->Bind (InetSocketAddress (80));
   dr);
src.sin port = htons(80);
bind(sk, (struct sockaddr *) &src,
 sizeof(src));
struct sockaddr in dest;
                                            sk->SendTo (InetSocketAddress (Ipv4Address
inet pton(AF INET, "10.0.0.1", &dest.sin
                                               ("10.0.0.1"), 80), Create<Packet>
                                               ("hello", 6));
   addr);
dest.sin port = htons(80);
sendto(sk, "hello", 6, 0, (struct
   sockaddr *) &dest, sizeof(dest));
char buf[6];
                                            sk->SetReceiveCallback (MakeCallback
recv(sk, buf, 6, 0);
                                               (MySocketReceive));
                                            • [...] (Simulator::Run ())
                                            void MySocketReceive (Ptr<Socket> sk,
                                               Ptr<Packet> packet)
                                                                                    62
                                         ns-3
```

ns-3 tutorial agenda

• 13h00-15h00: Getting started with ns-3

- Overview of software and models
- Basic structure of the core and important models
- 15h00-15h30: 30-minute coffee break
- 15h40-17h15: Going further with ns-3
 - Running and understanding an existing example
 - Animation and visualization
 - Writing and debugging your own examples
 - Integrating other tools and libraries
 - Parallel simulations
 - Emulation, virtual machine and testbed integration
 - Getting help and getting involved



Attributes and default values

```
// disable fragmentation for frames below 2200 bytes
 Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold", StringValue ("22
00"));
 // turn off RTS/CTS for frames below 2200 bytes
 Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue ("2200"));
 // Fix non-unicast data rate to be the same as that of unicast
 Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",
                     StringValue (phyMode));
 NodeContainer c;
 c.Create (numNodes):
 // The below set of helpers will help us to put together the wifi NICs we want
 WifiHelper wifi;
 if (verbose)
   Ł
     wifi.EnableLogComponents (); // Turn on all Wifi logging
   }
 YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
 // set it to zero; otherwise, gain will be added
 wifiPhy.Set ("RxGain", DoubleValue (-10) );
 // ns-3 supports RadioTap and Prism tracing extensions for 802.11b
 wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT IEEE802 11 RADIO);
```



ns-3 attribute system

<u>Problem:</u> Researchers want to identify all of the values affecting the results of their simulations

- and configure them easily

<u>ns-3 solution:</u> Each ns-3 object has a set of attributes:

- A name, help text
- A type
- An initial value
- Control all simulation parameters for static objects
- Dump and read them all in configuration files
- Visualize them in a GUI
- Makes it easy to verify the parameters of a simulation



Short digression: Object metadata system

- ns-3 is, at heart, a C++ object system
- ns-3 objects that inherit from base class ns3::Object get several additional features
 - -dynamic run-time object aggregation
 - -an attribute system
 - smart-pointer memory management (Class Ptr)

We focus here on the attribute system



Use cases for attributes

- An Attribute represents a value in our system
- An Attribute can be connected to an underlying variable or function
 - -e.g. TcpSocket::m_cwnd;
 - -or a trace source



Use cases for attributes (cont.)

- What would users like to do?
 - Know what are all the attributes that affect the simulation at run time
 - -Set a default initial value for a variable
 - -Set or get the current value of a variable
 - Initialize the value of a variable when a constructor is called
- The attribute system is a unified way of handling these functions



- The traditional C++ way:
 - -export attributes as part of a class's public API
 - walk pointer chains (and iterators, when needed) to find what you need
 - -use static variables for defaults
- The attribute system provides a more convenient API to the user to do these things



Navigating the attributes

- Attributes are exported into a string-based namespace, with filesystem-like paths

 namespace supports regular expressions
- Attributes also can be used without the paths

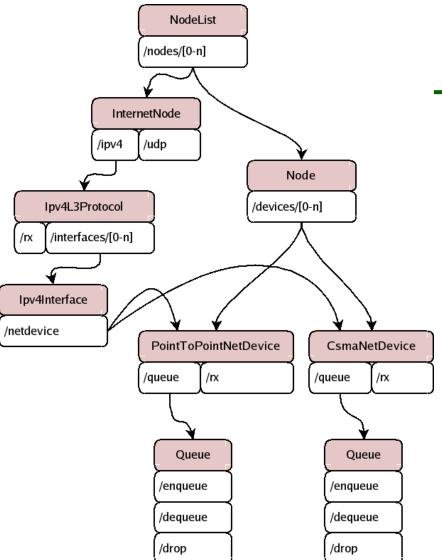
-e.g. "ns3::WifiPhy::TxGain"

• A Config class allows users to manipulate the attributes



Attribute namespace

 strings are used to describe paths through the namespace



Config::Set ("/NodeList/1/\$ns3::Ns3NscStack<linux2.6.26>/net.ipv4.tcp_sack", StringValue ("0"));



Navigating the attributes using paths

- Examples:
 - -Nodes with Nodelds 1, 3, 4, 5, 8, 9, 10, 11:

"/NodeList/[3-5]|[8-11]|1"

 UdpL4Protocol object instance aggregated to matching nodes:

"/\$ns3::UdpL4Protocol"



e.g.: Set a default initial value for a variable

Config::Set ("ns3::WifiPhy::TxGain", DoubleValue (1.0));

• Syntax also supports string values:

Config::Set ("WifiPhy::TxGain", StringValue
 ("1.0"));

Attribute

Value



Fine-grained attribute handling

- Set or get the current value of a variable
 - Here, one needs the path in the namespace to the right instance of the object
 - Config::SetAttribute("/NodeList/5/DeviceList/3/Ph
 y/TxGain", DoubleValue(1.0));

• Users can get Ptrs to instances also, and Ptrs to trace sources, in the same way



ns-3 attribute system

- Object attributes are organized and documented in the Doxygen
- Enables the construction of graphical configuration tools:

Object Attributes	Attribute Value
∽ ns3::NodeListPriv	
▽ NodeList	
▽ 0	
⊽ 0	
Address	00:00:00:00:00:01
EncapsulationMode	Llc
SendEnable	true
ReceiveEnable	true
DataRate	500000bps
▷ TxQueue	
▶ 1	
ApplicationList	
ns3::PacketSocketFactory	
▶ ns3::Ipv4L4Demux	
▷ ns3::Tcp	
ns3::Udp	
ns3::Ipv4	
ns3::ArpL3Protocol	
▷ ns3::Ipv4L3Protocol	



Attribute documentation

Main Page Related Pages Modules Name	espaces Classes Files
	The list of all attributes. [^{Core}]
Collaboration diagram for The list of all at	ributes.:
	Core The list of all attributes.

ns3::V4Ping

· Remote: The address of the machine we want to ping.

ns3::ConstantRateWifiManager

- DataMode: The transmission mode to use for every data packet transmission
- ControlMode: The transmission mode to use for every control packet transmission.

ns3::WifiRemoteStationManager

- IsLowLatency: If true, we attempt to modelize a so-called low-latency device: a device where decisions about tx parameters can be made on a per-packet basis and feedback about the transmission of each packet is obtained before sending the next. Otherwise, we modelize a high-latency device, that is a device where we cannot update our decision about tx parameters after every packet transmission.
- MaxSsrc: The maximum number of retransmission attempts for an RTS. This value will not have any effect on some rate control
 algorithms.
- MaxSIrc: The maximum number of retransmission attempts for a DATA packet. This value will not have any effect on some rate control algorithms.
- RtsCtsThreshold: If a data packet is bigger than this value, we use an RTS/CTS handshake before sending the data. This value
 will not have any effect on some rate control algorithms.



Options to manipulate attributes

- Individual object attributes often derive from default values
 - Setting the default value will affect all subsequently created objects
 - Ability to configure attributes on a per-object basis
- Set the default value of an attribute from the command-line: CommandLine cmd; cmd.Parse (argc, argv);
- Set the default value of an attribute with NS_ATTRIBUTE_DEFAULT
- Set the default value of an attribute in C++: Config::SetDefault ("ns3::Ipv4L3Protocol::CalcChecksum", BooleanValue (true));
- Set an attribute directly on a specic object:
 Ptr<CsmaChannel> csmaChannel = ...;
 csmaChannel->SetAttribute ("DataRate",
 StringValue ("5Mbps"));



Object names

- It can be helpful to refer to objects by a string name
 - -"access point"
 - -"eth0"
- Objects can now be associated with a name, and the name used in the attribute system



Names example

```
NodeContainer n;
n.Create (4);
Names::Add ("client", n.Get (0));
Names::Add ("server", n.Get (1));
...
Names::Add ("client/eth0", d.Get (0));
...
Config::Set ("/Names/client/eth0/Mtu", UintegerValue
(1234));
```

Equivalent to:

Config::Set ("/NodeList/0/DeviceList/0/Mtu", UintegerValue
 (1234));



Tracing and statistics

- Tracing is a structured form of simulation output
- Example (from ns-2):
- + 1.84375 0 2 cbr 210 ----- 0 0.0 3.1 225 610
- 1.84375 0 2 cbr 210 ----- 0 0.0 3.1 225 610
- r 1.84471 2 1 cbr 210 ----- 1 3.0 1.0 195 600
- r 1.84566 2 0 ack 40 ----- 2 3.2 0.1 82 602
- + 1.84566 0 2 tcp 1000 ----- 2 0.1 3.2 102 611

Problem: Tracing needs vary widely

- -would like to change tracing output without editing the core
- -would like to support multiple outputs

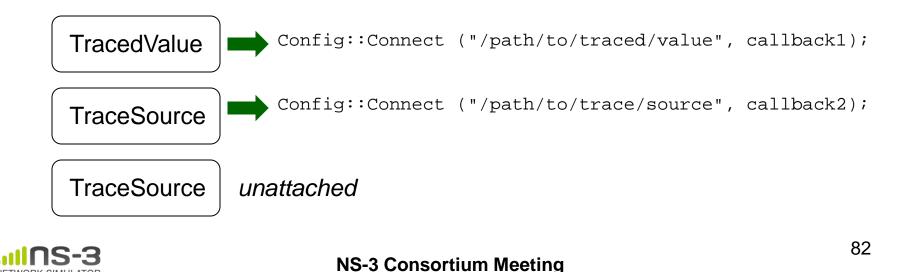


- Simulator provides a set of pre-configured trace sources
 - -Users may edit the core to add their own
- Users provide trace sinks and attach to the trace source
 - Simulator core provides a few examples for common cases
- Multiple trace sources can connect to a trace sink



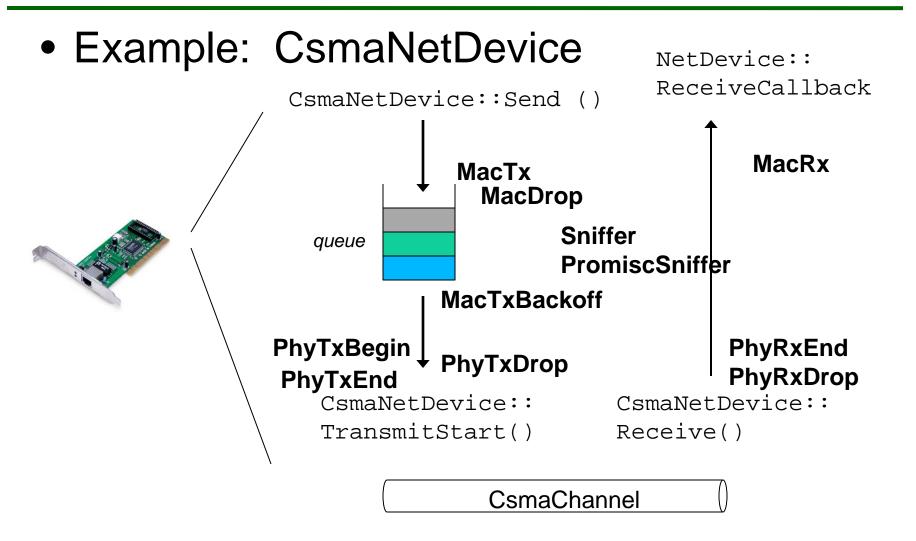
Tracing in ns-3

- ns-3 configures multiple 'TraceSource' objects (TracedValue, TracedCallback)
- Multiple types of 'TraceSink' objects can be hooked to these sources
- A special configuration namespace helps to manage access to trace sources



March 2013

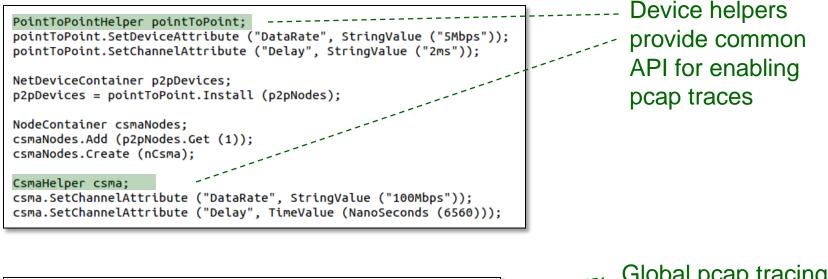
NetDevice trace hooks

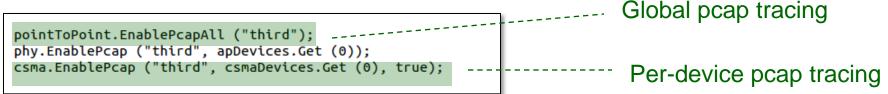




Enabling tracing in your code

• examples/tutorial/third.cc







Discovering ns-3 trace sources

- various trace sources (e.g., packet receptions, state machine transitions) are plumbed through the system
- Organized with the rest of the attribute system

NS-3	Main Page Modules Namespaces Classes Related Pages			
ns-3 Documentation	The list of all trace courses			
NS-3 Modules	The list of all trace sources.			
NS-3 Class List	[Core]			
NS-3 Class Hierarchy	Collaboration diagram for The list of all trace sources.:			
🕂 🗐 Class Members				
NS-3 Graphical Class Hierarchy	Core The list of all trace sources.			
👒 NS-3 Namespace List	ns3::WifiNetDevice			
🗉 🖹 Namespace Members				
NS-3 Related Pages	 Rx: Received payload from the MAC layer. Tx: Send payload to the MAC layer. 			
	ns3::WifiPhy			
	State: The WifiPhy state			
	 RxOk: A packet has been received successfully. 			
	 RxError: A packet has been received unsuccessfully. 			
	Tx: Packet transmission is starting.			
	ns3::MobilityModel			
	 CourseChange: The value of the position and/or velocity vector changed 			
	ns3::olsr::Agentimpi			
	Rx: Receive OLSR packet.			
	Tx: Send OLSR packet.			
	 RoutingTableChanged: The OLSR routing table has changed. 			



ns3::PacketSink

Basic tracing

 Helper classes hide the tracing details from the user, for simple trace types

 ascii or pcap traces of devices

```
if (tracing == true)
{
    AsciiTraceHelper ascii;
    wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("wifi-simple-adhoc-grid.tr"));
    wifiPhy.EnablePcap ("wifi-simple-adhoc-grid", devices);
    // Trace routing tables
    Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ("wifi-simple-adhoc-grid.routes", std::ios::out);
    olsr.PrintRoutingTableAllEvery (Seconds (2), routingStream);
    // To do-- enable an IP-level trace that shows forwarding events only
}
```



- Highest-level: Use built-in trace sources and sinks and hook a trace file to them
- Mid-level: Customize trace source/sink behavior using the tracing namespace
- Low-level: Add trace sources to the tracing namespace

-Or expose trace source explicitly



• Highest-level: Use built-in trace sources and sinks and hook a trace file to them

// Also configure some tcpdump traces; each interface will be traced

- // The output files will be named
- // simple-point-to-point.pcap-<nodeId>-<interfaceId>
- // and can be read by the "tcpdump -r" command (use "-tt" option to
- // display timestamps correctly)

PcapTrace pcaptrace ("simple-point-to-point.pcap");

pcaptrace.TraceAllIp ();





• Mid-level: Customize trace source/sink behavior using the tracing namespace



Asciitrace: under the hood



}

Lowest-level of tracing

• Low-level: Add trace sources to the tracing namespace

Config::Connect ("/NodeList/.../Source",

MakeCallback (&ConfigTest::ChangeNotification, this));



Review of topics covered

- Structure of an ns-3 program
- Fundamental classes

 Nodes, NetDevices, Channels, Applications
- Node and device containers
- Helper APIs, and Install pattern
- Wifi and Internet stack architecture
- Attributes and default values
- Tracing



Animation and visualization



FlowMonitor

- Network monitoring framework found in src/flow-monitor/
- Goals:
 - -detect all flows passing through network
 - stores metrics for analysis such as bitrates, duration, delays, packet sizes, packet loss ratios

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



FlowMonitor architecture

- Basic classes
 - FlowMonitor
 - FlowProbe
 - FlowClassifier
 - FlowMonitorHelper
- Ipv4 only

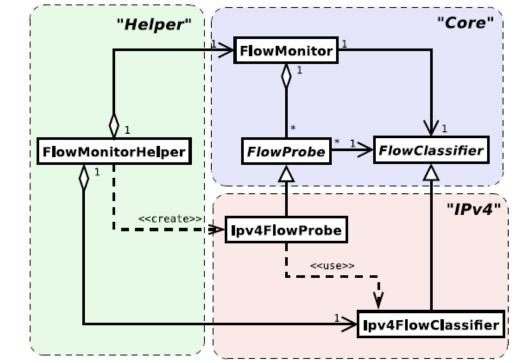


Figure credit: G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.



FlowMonitor statistics

Statistics gathered

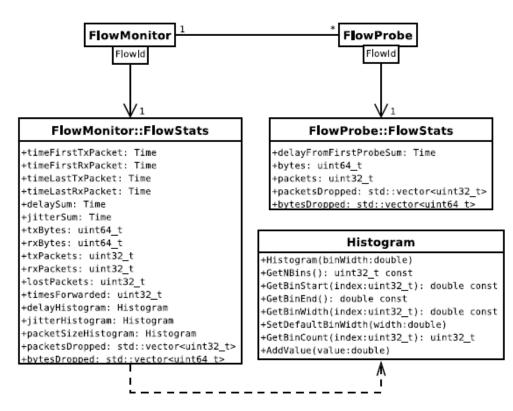


Figure credit: G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.



FlowMonitor configuration

• example/wireless/wifi-hidden-terminal.cc

```
// 8. Install FlowMonitor on all nodes
FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll ();
// 9. Run simulation for 10 seconds
Simulator::Stop (Seconds (10));
Simulator::Run ();
// 10. Print per flow statistics
monitor->CheckForLostPackets ();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
for (std::map<FlowId, FlowMonitor::FlowStats>::const iterator i = stats.begin (); i != stats.end (); ++i)
  {
    // first 2 FlowIds are for ECHO apps, we don't want to display them
   if (i - first > 2)
     ſ
        Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
        std::cout << "Flow " << i->first - 2 << " (" << t.sourceAddress << " -> " << t.destinationAddress << ")\n";</pre>
        std::cout << " Tx Bytes: " << i->second.txBytes << "\n";</pre>
        std::cout << " Rx Bytes: " << i->second.rxBytes << "\n";</pre>
        std::cout << " Throughput: " << i->second.rxBytes * 8.0 / 10.0 / 1024 / 1024 << " Mbps\n";</pre>
```



FlowMonitor output

- This program exports statistics to stdout
- Other examples integrate with PyViz

```
Hidden station experiment with RTS/CTS disabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
 Tx Bytes:
              3847500
  Rx Bytes:
              316464
  Throughput: 0.241443 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:
              3848412
  Rx Bvtes:
              336756
 Throughput: 0.256924 Mbps
Hidden station experiment with RTS/CTS enabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
 Tx Bvtes:
              3847500
  Rx Bytes:
              306660
  Throughput: 0.233963 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes:
              3848412
  Rx Bvtes:
              274740
  Throughput: 0.20961 Mbps
```

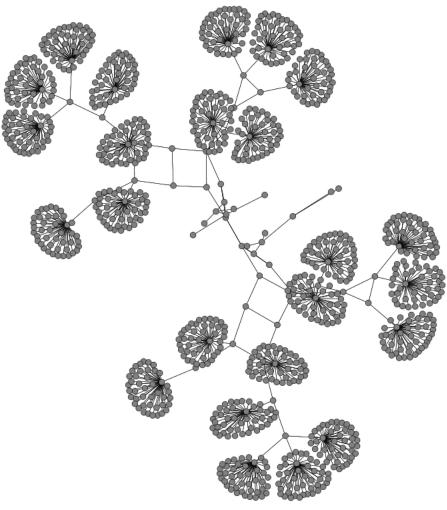


PyViz overview

- Developed by Gustavo Carneiro
- Live simulation visualizer (no trace files)
- Useful for debugging – mobility model behavior
 - -where are packets being dropped?
- Built-in interactive Python console to debug the state of running objects
- Works with Python and C++ programs



Pyviz screenshot (Graphviz layout)

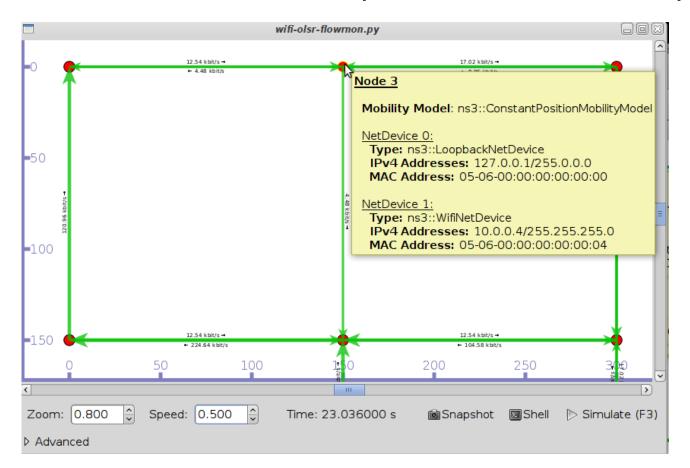






Pyviz and FlowMonitor

src/flow-monitor/examples/wifi-olsr-flowmon.py





Enabling PyViz in your simulations

Make sure PyViz is enabled in the build

SQlite stats data output	: not enabled (library 'sqlite3' not found)
Tap Bridge	: enabled
PyViz visualizer	: enabled
Use sudo to set suid bit	: not enabled (optionenable-sudo not selected)

 If program supports CommandLine parsing, pass the option

--SimulatorImplementationType=

ns3::VisualSimulatorImpl

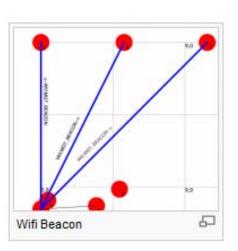
• Alternatively, pass the "--vis" option

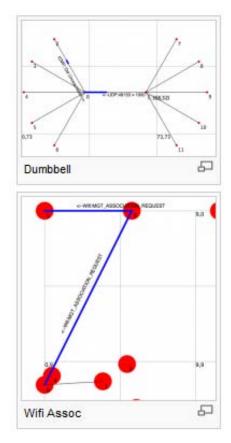


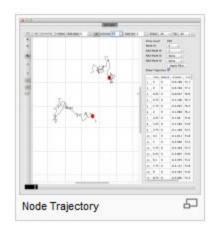
NetAnim

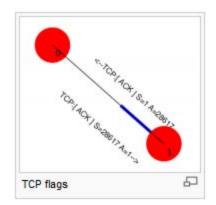
"NetAnim" by George Riley and John Abraham

	To Person	FORT RADIE OF	11.0044.00	
١.	25+-85	8	\$	weiect_stacte franks alsos o be entered
١.	210-01	8	8	WEINCH, MACON THE DE DINOLD DIN BUT MIT
١.	2.5+-85	8	9	whiteCLEACEN Funds: 1 ups of a first first
۰.	8.066187538	5	6	WE NOT ASSOCIATION, BID 451 Frends, 8 w/dt.
ŧ.	8.080187025	5		WEINCH, ASSOCIATION, INC. ANT Franchis, IT SUPE-
ι.	8-040187037	5	0	W8 INCLASSOCIATION, 80(0.857 Franklin, 8:000
١.	0.000178086	8	8	WA CTS, ACK MAINS BO IN SHIELD
١.	100079066		6	WE CILLAD BAROR REDIK REDIK
ŧ.	8.000279040	8	r	WH (75, ADX 84.80-80-08-08-08-08-0
ii.	8.0004403388	6	4	WEIRCT, ASSOCIATION, RE2, AS7 RoardS, E1605.
ñ.	6.006462183	6	6	wit HCT_ASSOCATION_REQ.EST Franklik is not
ia,	8.08011414		1	WE CTL ACK MAND IN DRIVE OF DRIVE
12	8.00051414	8	6	WE CTLACK MAND TO BE DRIVE OF DR
i.	8.00011414	8	3	WE CTL ADV BAIRS OF BEIDE OF DE











NetAnim key features

- Animate packets over wired-links and wirelesslinks
 - limited support for LTE traces
- Packet timeline with regex filter on packet metadata.
- Node position statistics with node trajectory plotting (path of a mobile node).
- Print brief packet-meta data on packets



Writing and debugging your own examples



Writing and debugging new programs

- Choosing between Python and C++
- Reading existing code
- Understanding and controlling logging code
- Error conditions
- Running programs through a debugger



Python bindings

- ns-3 uses the 'pybindgen' tool to generate
 Python bindings for the underlying C++ libraries
- Existing bindings are typically found in the bindings/ directory of a module
- Some methods are not provided in Python (e.g. hooking trace sources)
- Generating new bindings requires a toolchain documented on the ns-3 web site



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 can be done, but callbacks and templates make it more challenging than usual



Debugging support

- Assertions: NS_ASSERT (expression);
 - Aborts the program if expression evaluates to false
 - Includes source file name and line number
- Unconditional Breakpoints: NS_BREAKPOINT ();
 - Forces an unconditional breakpoint, compiled in
- Debug Logging (not to be confused with tracing!)
 - Purpose
 - Used to trace code execution logic
 - For debugging, not to extract results!
 - Properties
 - NS_LOG* macros work with C++ IO streams
 - E.g.: NS_LOG_UNCOND ("I have received " << p->GetSize () << " bytes");
 - NS_LOG macros evaluate to nothing in optimized builds
 - When debugging is done, logging does not get in the way of execution performance



Debugging support (cont.)

- Logging levels:
 - NS_LOG_ERROR (...): serious error messages only
 - NS_LOG_WARN (...): warning messages
 - NS_LOG_DEBUG (...): rare ad-hoc debug messages
 - NS_LOG_INFO (...): informational messages (eg. banners)
 - NS_LOG_FUNCTION (...):function tracing
 - NS_LOG_PARAM (...): parameters to functions
 - NS_LOG_LOGIC (...): control flow tracing within functions
- Logging "components"
 - Logging messages organized by components
 - Usually one component is one .cc source file
 - NS_LOG_COMPONENT_DEFINE ("OlsrAgent");
- Displaying log messages. Two ways:
 - Programatically:
 - LogComponentEnable("OlsrAgent", LOG_LEVEL_ALL);
 - From the environment:
 - NS_LOG="OlsrAgent" ./my-program



Running C++ programs through gdb

- The gdb debugger can be used directly on binaries in the build directory
- An easier way is to use a waf shortcut
 - ./waf --command-template="gdb %s" --run <programname>
- Note: valgrind can be run similarly

./waf --command-template="valgrind %s" --run
<program-name>



Testing

- Can you trust ns-3 simulations?
 - Can you trust *any* simulation?
 - Onus is on the simulation project to validate and document results
 - Onus is also on the researcher to verify results
- ns-3 strategies:
 - regression and unit tests
 - Aim for event-based rather than trace-based
 - validation of models on testbeds
 - reuse of code



Test framework

- ns-3-dev is checked nightly on multiple platforms
 - Linux gcc-4.x, i386 and x86_64, OS X i386, FreeBSD and Cygwin (occasionally)
- ./test.py will run regression tests

Walk through test code, test terminology (suite, case), and examples of how tests are run



Improving performance

- Debug vs optimized builds
 - ./waf -d debug configure
 - ./waf -d debug optimized
- Build ns-3 with static libraries
 - ./waf --enable-static
- Use different compilers (icc)
 - has been done in past, not regularly tested



Integrating other tools and libraries



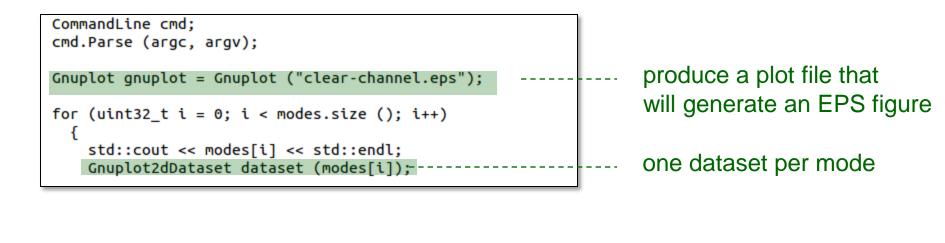
Gnuplot

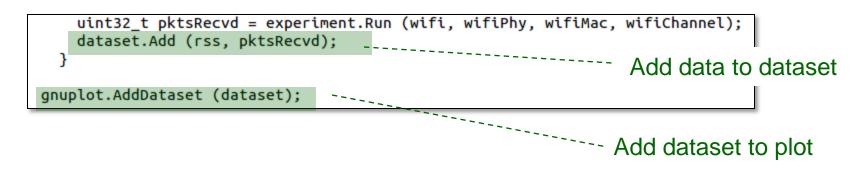
- src/tools/gnuplot.{cc,h}
- C++ wrapper around gnuplot
- classes:
 - -Gnuplot
 - -GnuplotDataset
 - Gnuplot2dDataset, Gnuplot2dFunction
 - Gnuplot3dDataset, Gnuplot3dFunction



Enabling gnuplot for your code

• examples/wireless/wifi-clear-channel-cmu.cc









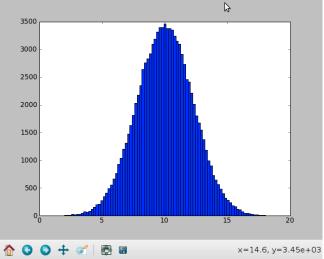
• src/core/examples/sample-rng-plot.py

```
# 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()
```





Other libraries

- ns-3 supports additional libraries (click, openflow, nsc)
- ns-3 has optional libraries (libxml2, gsl, mysql)
- both are typically enabled/disabled through the wscript
- users are free to write their own Makefiles or wscripts to do something special



Scaling to multiple machines



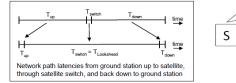
Overview

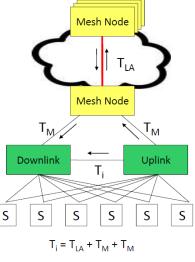
- Parallel and distributed discrete event simulation
 - Allows single simulation program to run on multiple interconnected processors
 - Reduced execution time! Larger topologies!
- Terminology
 - Logical process (LP)
 - Rank or system id



Simulation size record

- Simulation on a HPC cluster at the U.S. Mobile Network Modeling Institute (2011) *
 - 176 cores, 3 TB of memory
 - 360,448,000 simulated nodes
 - 413,704.52 packet receive events per second [wall-clock]
 - Each NS-3 Federate only instantiates its own subnets, a "mesh" node, and stubs for external mesh nodes
 - Number of PTP links: $N^2 \rightarrow N$
 - Requires interface re-numbering
 - Static routing for unicast & multicast
 - Inter-federate latency maximized
 - Time from uplink and downlink latencies and moved into satellite switching latency
 - Intra-federate latency between subnets matches inter-federate latency
 - Scenario run for 20 minutes of simulated time

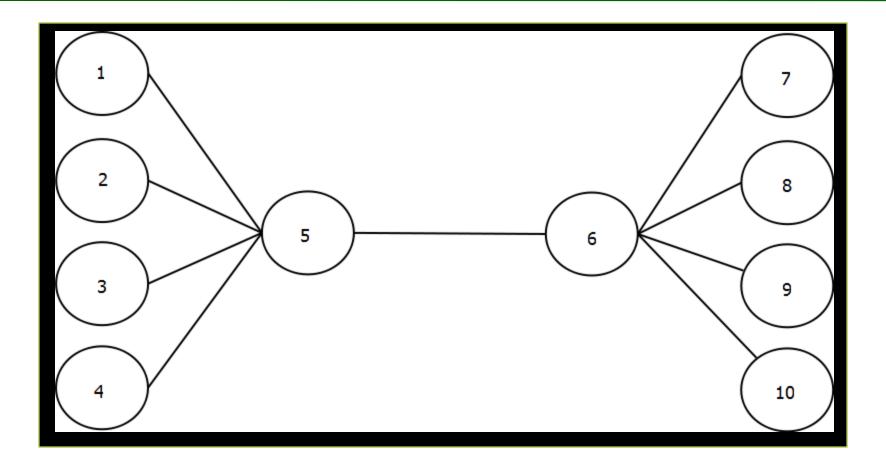




* K. Renard et al, "A Performance and Scalability Evaluation of the NS-3 Distributed Scheduler. Proceedings of WNS3 2012.

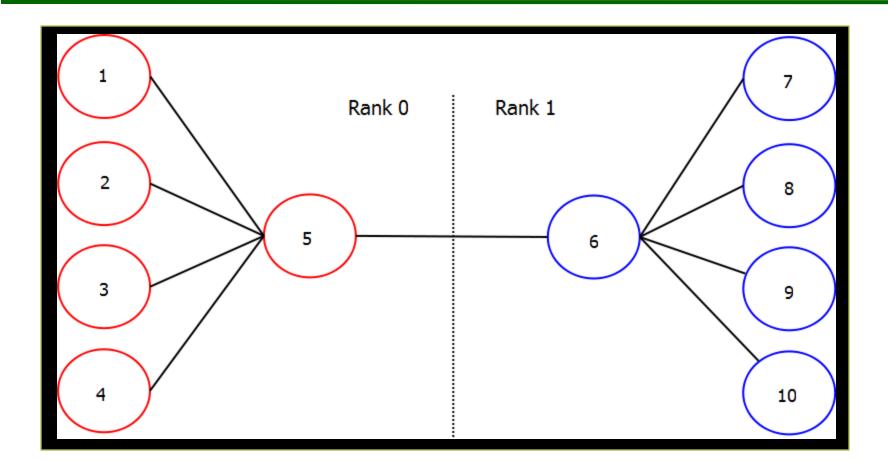


Quick and Easy Example





Quick and Easy Example





Implementation Details

- LP communication
 - Message Passing Interface (MPI) standard
 - Send/Receive time-stamped messages
 - MpiInterface in ns-3
- Synchronization
 - Conservative algorithm using lookahead
 - DistributedSimulator in ns-3



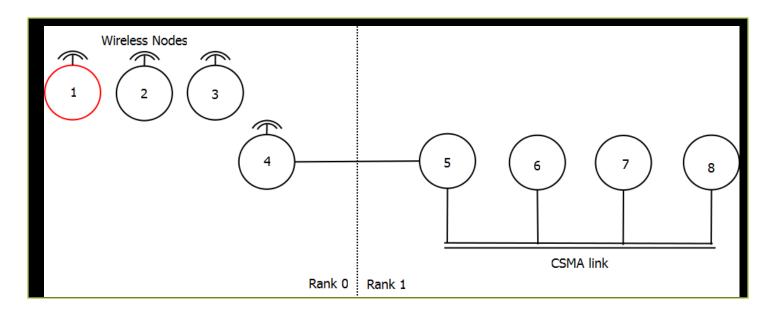
Implementation Details (cont.)

- Assigning rank
 - Currently handled manually in simulation script
 - Next step, MpiHelper for easier node/rank mapping
- Remote point-to-point links
 - Created automatically between nodes with different ranks through point-to-point helper
 - Packet sent across using MpiInterface



Implementation Details (cont.)

- Distributing the topology
 - -All nodes created on all LPs, regardless of rank
 - Applications are only installed on LPs with target node



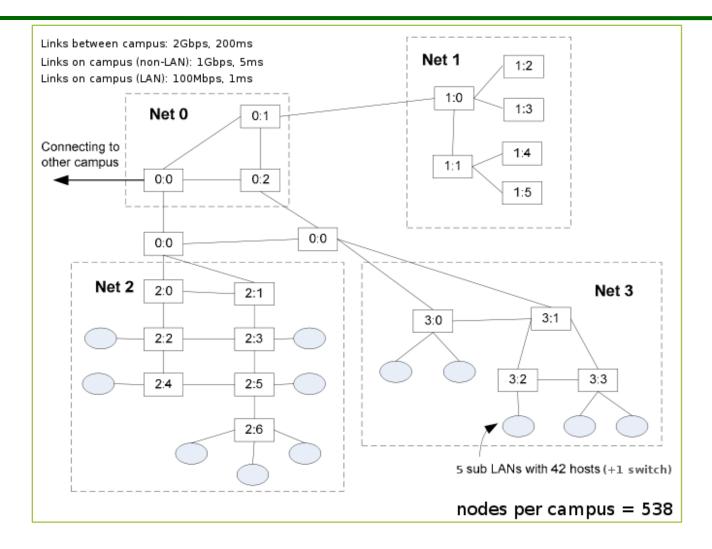


Performance Test

- DARPA NMS campus network simulation
 - -Allows creation of very large topologies
 - Any number of campus networks are created and connected together
 - Different campus networks can be placed on different LPs
 - Tested with 2 CNs, 4 CNs, and 6 CNs

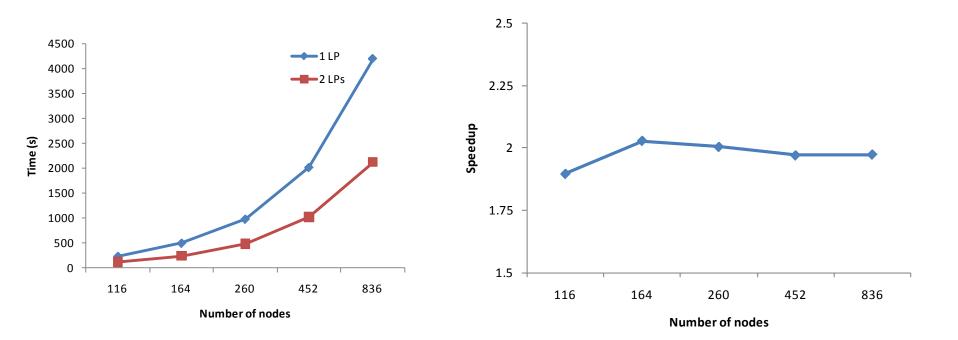


Campus Network Topology





2 Campus Networks





Summary

- Distributed simulation in ns-3 allows a user to run a single simulation in parallel on multiple processors
- By assigning a different rank to nodes and connecting these nodes with point-to-point links, simulator boundaries are created
- Simulator boundaries divide LPs, and each LP can be executed by a different processor
- Distributed simulation in ns-3 offers solid performance gains in time of execution for large topologies



emulation and testbeds

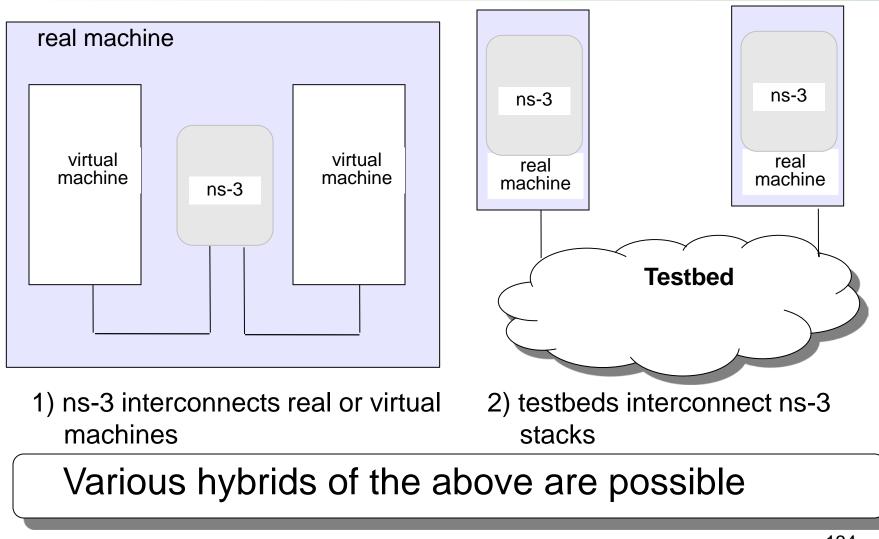


Emulation support

- Support moving between simulation and testbeds or live systems
- A real-time scheduler, and support for two modes of emulation
 - GlobalValue::Bind ("SimulatorImplementationType", StringValue ("ns3::RealTimeSimulatorImpl"));

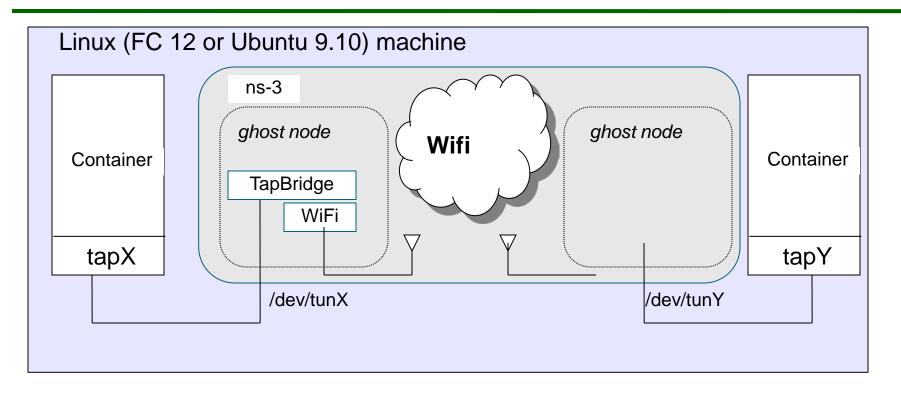


ns-3 emulation modes



NETWORK SIMULATOR

"Tap" mode: netns and ns-3 integration



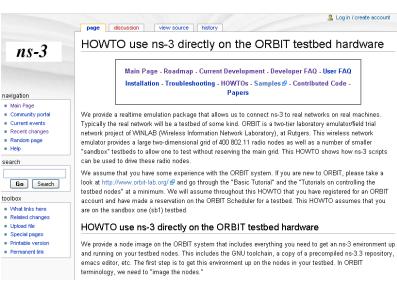
Tap device pushed into namespaces; no bridging needed



Example: ORBIT and ns-3

Support for use of Rutgers WINLAB ORBIT radio grid







Issues

- 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

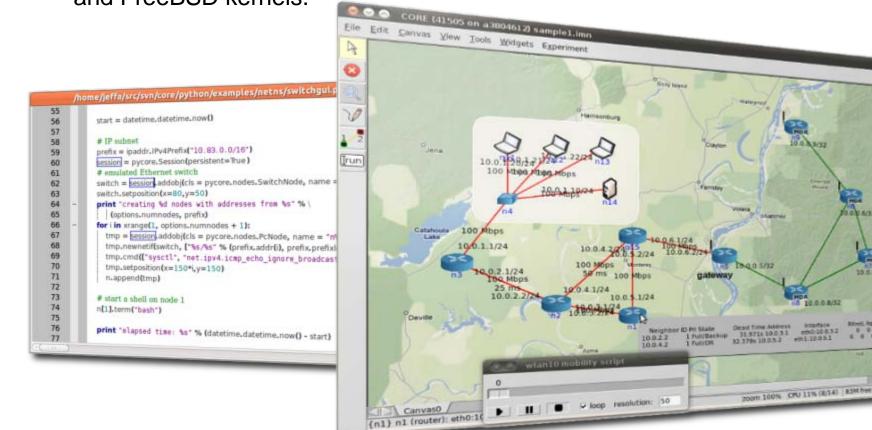


Container-based virtual machines and ns-3



What is CORE?

The Common Open Research Emulator (CORE) is a Python framework and GUI for emulating networks using lightweight Virtualization native to Linux and FreeBSD kernels.





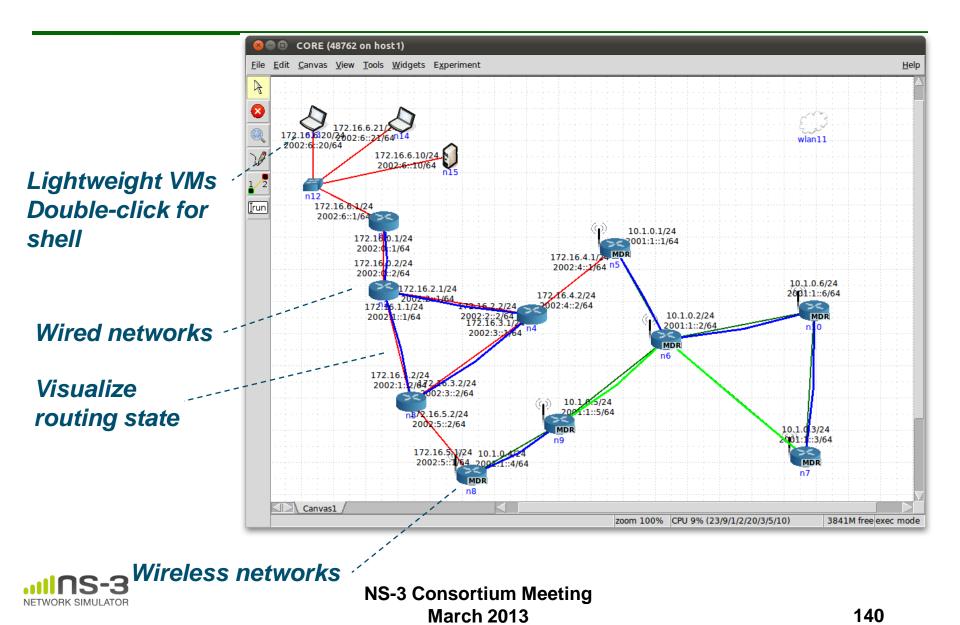
NS-3 Consortium Meeting March 2013

0.0 6/92

10.0.0 7/32

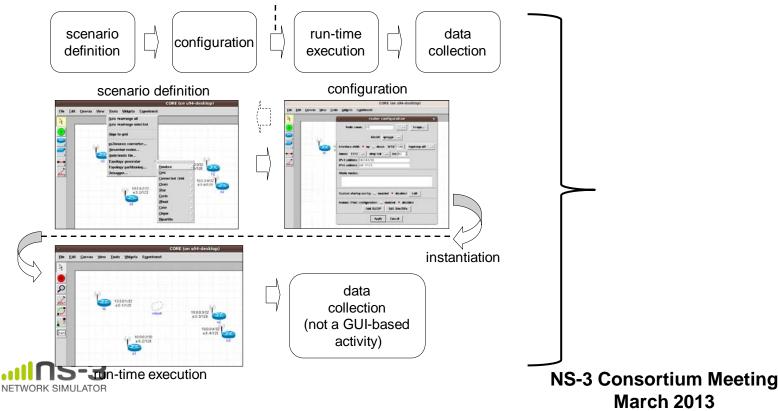
Alterit, Rgott, Deper

Screenshot



Technical Goals

- CORE provides Python libraries for using Linux network namespaces in network emulation experiments
 - CORE + ns-3 integrates realism of namespace with wireless device models
- CORE is a graphical controller that users find intuitive
 - CORE GUI could exentually be used for ns-3 authoring/visualization

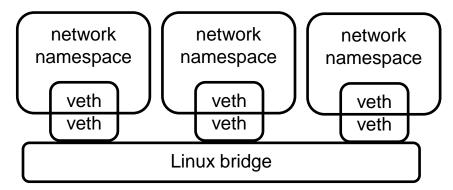


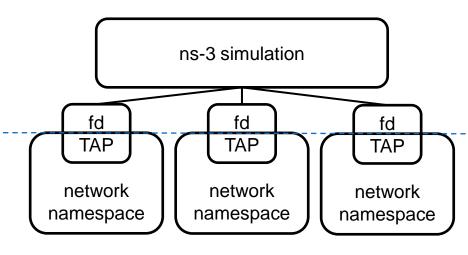
Virtual Interfaces

- Ordinary CORE
 - Virtual Ethernet pairs (veth) are installed into a namespace and joined to a bridge.
 - For wireless networks (WLANs), ebtables rules govern pairwise connectivity.



- TUN/TAP device installed into a namespace, socket held by _ simulation.
- Simulation runs with real-time scheduler.







Mobility demonstration

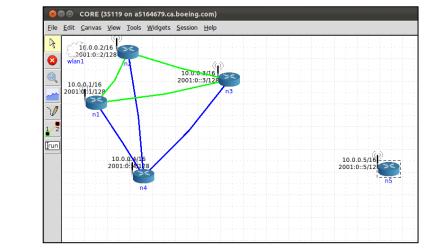
Canvas-based mobility

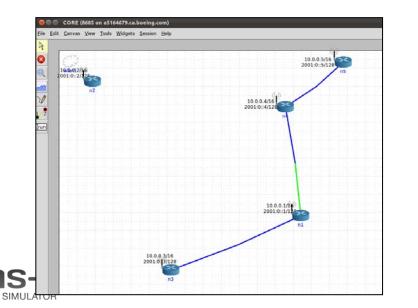
ns-3 ConstantPosition

MobilityModel

users can drag nodes
around and change

topology





ns-3 mobility visualizationns-3 RandomWalk MobilityModel

• users can observe Linux namespace state (e.g. OSPF adjacencies) as nodes move in the ns-3 realm

Scaling time in virtualized environments

- Synchronized Network Emulation RWTH Aachen University
 - Modified Xen
- VAN Testbed Telcordia/CERDEC
 Modified Xen
- Linux Time namespace Jeff Dike (UML creator)
 - Add a time namespace to the Linux kernel, allowing for gettimeofday() offsets



Direct Code Execution



Goals

- Lightweight virtualization of kernel and application processes, interconnected by simulated networks
- Benefits:
 - Implementation realism in controlled topologies or wireless environments
 - Model availability
- Limitations:
 - Not as scalable as pure simulation
 - Runs in real-time
 - Integration of the two environments



Direct Code Execution

- Developed by Mathieu Lacage and Frederic Urbani, INRIA, Hajime Tazaki (WIDE)
- Run unmodified application binaries in ns-3
 - Also, can run entire Linux stack in ns-3

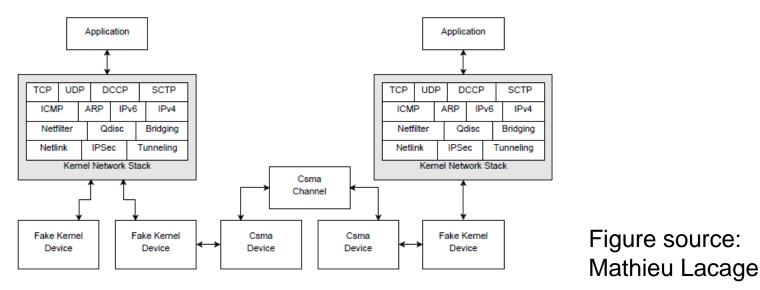


Figure 4.6: The Linux network stack running inside ns-3

http://www-sop.inria.fr/members/Frederic.Urbani/ns3dceccnx/index.html







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

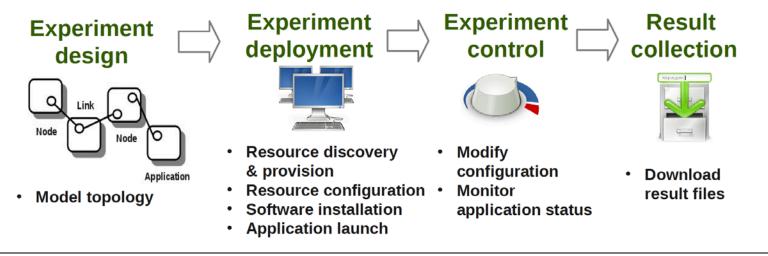


Figure source: Alina Quereilhac, INRIA



Getting Help and Getting Involved



Resources

Web site:

http://www.nsnam.org

Mailing list:

http://mailman.isi.edu/mailman/listinfo/ns-developers

IRC: #ns-3 at freenode.net

Tutorial:

http://www.nsnam.org/docs/tutorial/tutorial.html

Code server:

http://code.nsnam.org

Wiki:

http://www.nsnam.org/wiki/index.php/Main_Page



Questions?

