Intelligent Wireless Network Group
Department of Computer Engineering
Faculty of Engineering, Kasetsart University
http://iwing.cpe.ku.ac.th

ns-3 Tutorial (Part I) Introduction

JCSSE 2011's tutorials and workshops
Wednesday, 11 May 2011, 9:00 - 16:00

Instructors

- **Intelligent Wireless Network Group (IWING)**

  Department of Computer Engineering

  Kasetsart University

  http://iwing.cpe.ku.ac.th

รศ.ดร.อนันต์ ผลเพิ่ม

Assoc.Prof. Anan Phonphoem, Ph.D.

ผศ.ดร.ชัยพร ใจแก้ว

Asst.Prof. Chaiporn Jaikaeo, Ph.D.

อ.อภิรักษ์ จันทร์สร้าง

Aphirak Jansang, M.Eng. (Pursuing Ph.D.)

Time Table

- 09:00 - 10:15 ns-3 Introduction & Installation
- 10:15 - 10.30 Break
- 10:30 - 12:00 Hands-On:
  Point-to-point and CSMA (Ethernet)
- 12:00 - 13:00 Lunch
- 13:00 - 14:15 Hands-On:
  Wireless & Tracing System and Visualizing Results
- 14:15 - 14:30 Break
- 14:30 - 15:30 Demonstation:
  ns-3 protocol stack modification
- 15:30 - 16:00 Q&A

Credits

- Modified from ns-3 Tutorial (20 August 2010)
- Tutorial examples from www.nsnam.org
- Slides from
  - Gustavo J. A. M. Carneiro (Universidade do Porto)
  - George Riley (Georgia Institute of Technology)
  - Joe Kopena (Drexel University)
  - Other contributors ….

Examples of Available Simulation Tools

- **ns-2**
  - Original “design” by Steve McCanne
  - TCP/C++ Hybrid
  - Open Source (Numerous contributions)
  - De-facto Standard in Academic Research
- **OPNET**
  - Commercial, closed source tool
  - De-facto standard in Military and DoD programs
  - Full-Featured, nice GUI
  - Sophisticated Data Analysis features

Examples of Available Simulation Tools

- **Qualnet**
  - Commercial, closed source
  - Competes primarily with OPNET
    - Strengths are in wireless models and protocols
    - Scalability
  - Based on public-domain "GloMoSim" tools
- **OMNet++**
  - C++ engine
  - Very popular in European Community

ns-2: *Ancestor of ns-3*

- “Over 50% of ACM and IEEE network simulation papers from 2000-2004 cite the use of ns-2”
- Went unmaintained for a long period of time
- Outdated code design
- Tracing system is difficult to use (mostly printf)
- Need to parse trace files to extract results

ns-3 Overview

- Network Simulator version 3
- Rewritten from scratch
- An **open source** discrete event simulator
  - Events model packet transmission, receipt, timers, etc.
  - Future events maintained in sorted *Event List*
  - Processing events results in zero or more new events
- Target for networking research and education

NS-3 Overview

- Project started around mid 2006
  - Still under heavy development
- Partially funded by US NSF “Community Resource Initiative (CRI) grant
- Official funded partners:
  - University of Washington
    - (Tom Henderson, Craig Dowell)
  - INRIA, Sophia Antipolis
    - (Mathieu Lacage)
  - Georgia Tech University (Atlanta)
    - George Riley (main author of GTNetS)
    - Raj Bhattacharjea

ns-3 Basic

- Written completely in C++
  - Heavy use of Templates
  - C++ Namespace (ns3)
- Simulation programs are C++ executables
- Python bindings for public API's provided
- NS-3 uses the "waf" build system
  - Instead of `./configure; make` use
  - `./waf`
- Builds a dynamic library
  - Both a debug and optimized version

ns-3 Key Features

- Flexible Event Scheduler
  - Any member function on any object can be an event handler, with arbitrary parameter lists
- Trace output in ascii, or Pcap format
  - Use existing Pcap tools (eg. Wireshark)
- Numerous trace points enabled via callbacks
- Emulation mode
  - Integration with real networks/packets
  - Real-Time Scheduler
- Doxygen documentation
- Mercurial Code Repository
- Formal review/check-in procedure
- Quarterly releases

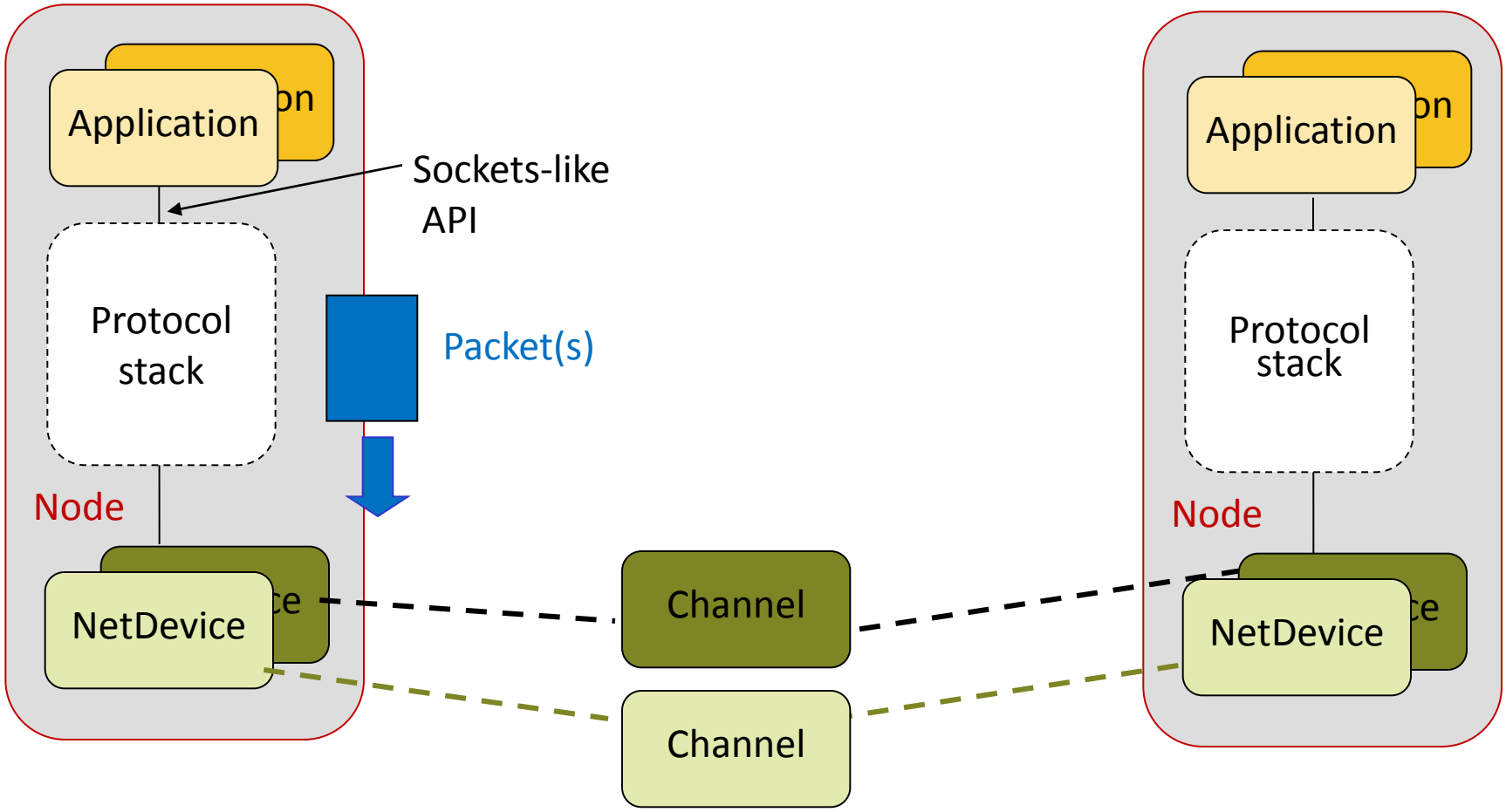
ns-3 Key Designs

- Use of "smart pointers" to ease memory management burden on code developers
- Use of "object aggregation", to allow extension of object functionality without adding additional virtual functions to base class.
  - Similar to Microsoft "Component Object Model"
- Integrated tracing framework based on type-safe callbacks
- Simulation event scheduling on arbitrary functions with arbitrary argument lists
- Packet objects manage sequential array of bytes with helper functions to add/remove headers and data

Components

- Node
- Net Device
- Channel
- Application
- Protocol Stack

The Basic ns-3 Data Flow Model

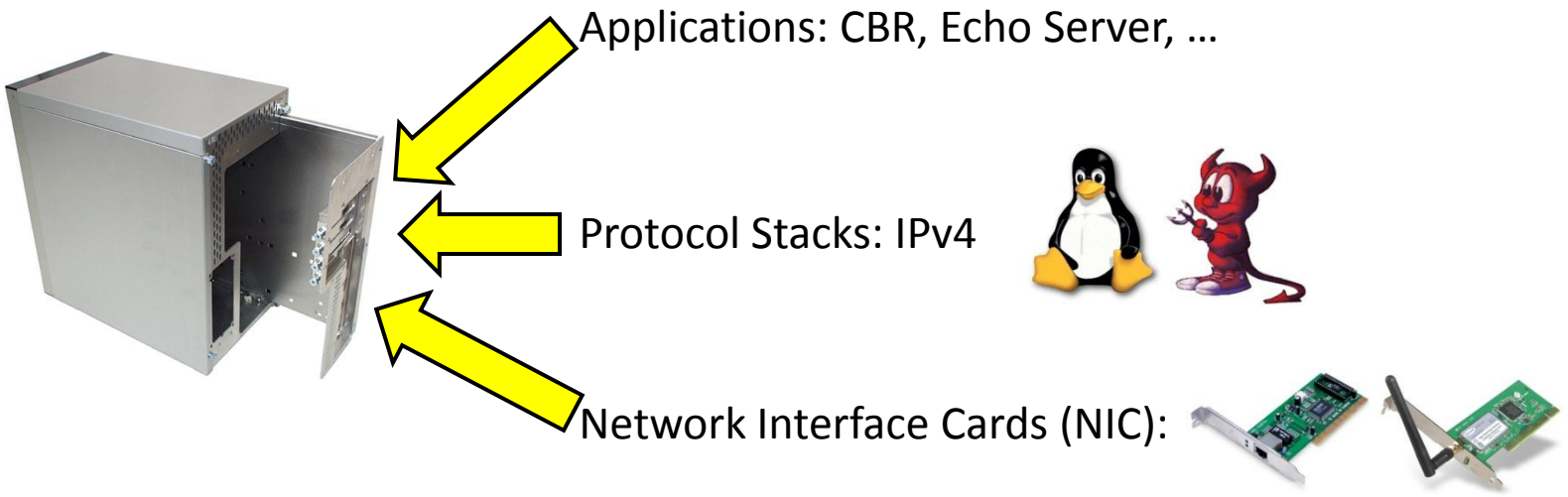


Architecture Elements

- Nodes may/may not have mobility, other traits
- Nodes have “network devices”
  - Network devices transfer packets over channels
  - Incorporating Layer 1 (Physical) & Layer 2 (Link)
- Devices interface w/ Layer 3 (Network: IP, ARP)
- Layer 3 supports Layer 4 (Transport: UDP, TCP)
- Layer 4 is used by Layer 5 (Application) objects

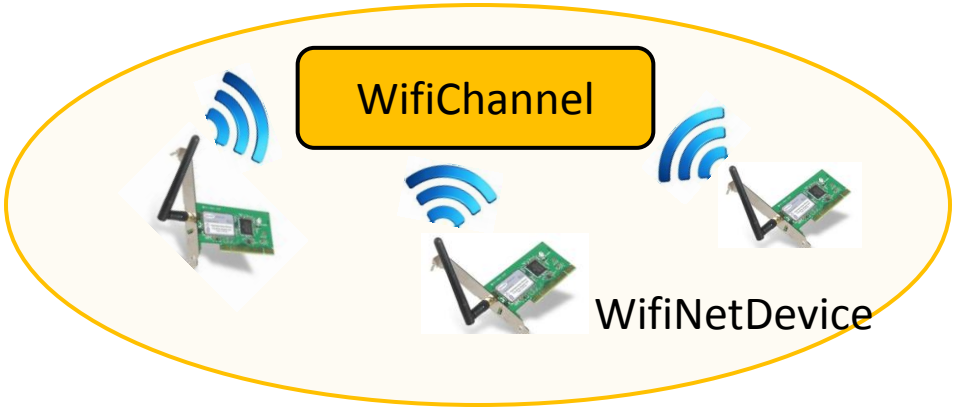
Node Structure

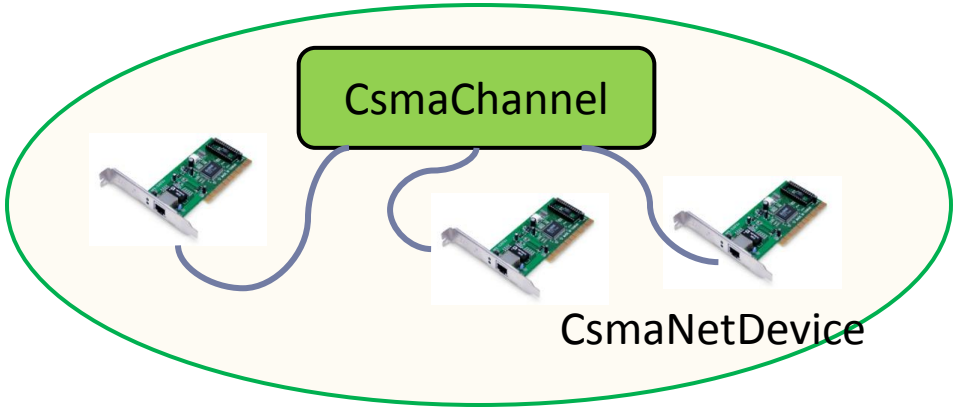
- A computer/ access point
- Installed NICs, protocol stacks, applications



Net Devices and Channels

- Net Devices are strongly bound to Channels of a matching type
- Net Devices example:
  - Ethernet NIC
  - WifiNetDevice
- Channel example:
  - CSMA Channel
  - WifiChannel





Link Layer Model

- Point-to-point (PPP links)
- Csma (Ethernet links)
- Bridge: 802.1D Learning Bridge
- Wifi (802.11 links)
  - EDCA QoS support (but not HCCA)
  - Both infrastructure (with beacons), and adhoc modes
- Mesh
  - 802.11s (but no legacy 802.11 stations supported yet)
  - ”Flame”: Forwarding LAyer for MEshing protocol
    - ”Easy Wireless: broadband ad-hoc networking for emergency services”
- Wimax: 802.16 (new in NS 3.8)
  - ”supports the four scheduling services defined by the 802.16-2004 standard”
- Tap-bridge, emu: testbed integration

Routing

- Adhoc:
  - OLSR (RFC 3626)
    - Since NS 3.8 with full HNA support (thanks Latih Suresh)
  - AODV (RFC 3561)
- ”Global routing” (aka GOD routing)
  - Just computes static routes on simulation start
- Nix-vector Routing
  - Limited but high performance static routing
  - For simulations with thousands of wired nodes
- List-routing
  - Joins multiple routing protocols in the same node
  - For example: static routing tables + OLSR + AODV

Applications (traffic generators)

- **Onoff**
  - Generates streams, alternating on-and-off periods
  - Highly parameterized
    - Can be configured to generate many types of traffic
      - E.g. OnTime=1 and OffTime=0 means CBR
    - Works with either UDP or TCP
- **Packet sink**: receives packets or TCP connnections
- **Ping6, v4ping**: send ICMP ECHO request
- **Udp-client/server**: sends UDP packet w/ sequence number
- **Udp-echo**: sends UDP packet, no sequence number
- **Radvd**: router advertisement (for IPv6)

ns-3 Packet

- each network packet contains a byte **buffer**, a list of **tags**, and **metadata**
  - **buffer:** bit-by-bit (serialized) representation of headers and trailers
  - **tags:** set of arbitrary, user-provided data structures (e.g., per-packet cross-layer messages, or flow identifiers)
  - **metadata:** describes types of headers and trailers that have been serialized
    - optional-- disabled by default
    - Enables packets to “print themselves”
- Implemented with Smart Pointers and Copy-on-Write Semantics

ns-3 Smart Pointers

- ns-3 uses reference-counting smart pointers at its APIs to limit memory leaks
  - Or “pass by value” or “pass by reference to const” where appropriate
- A “smart pointer” behaves like a normal pointer (syntax) but does not lose memory when reference count goes to zero
- Use them like built-in pointers:

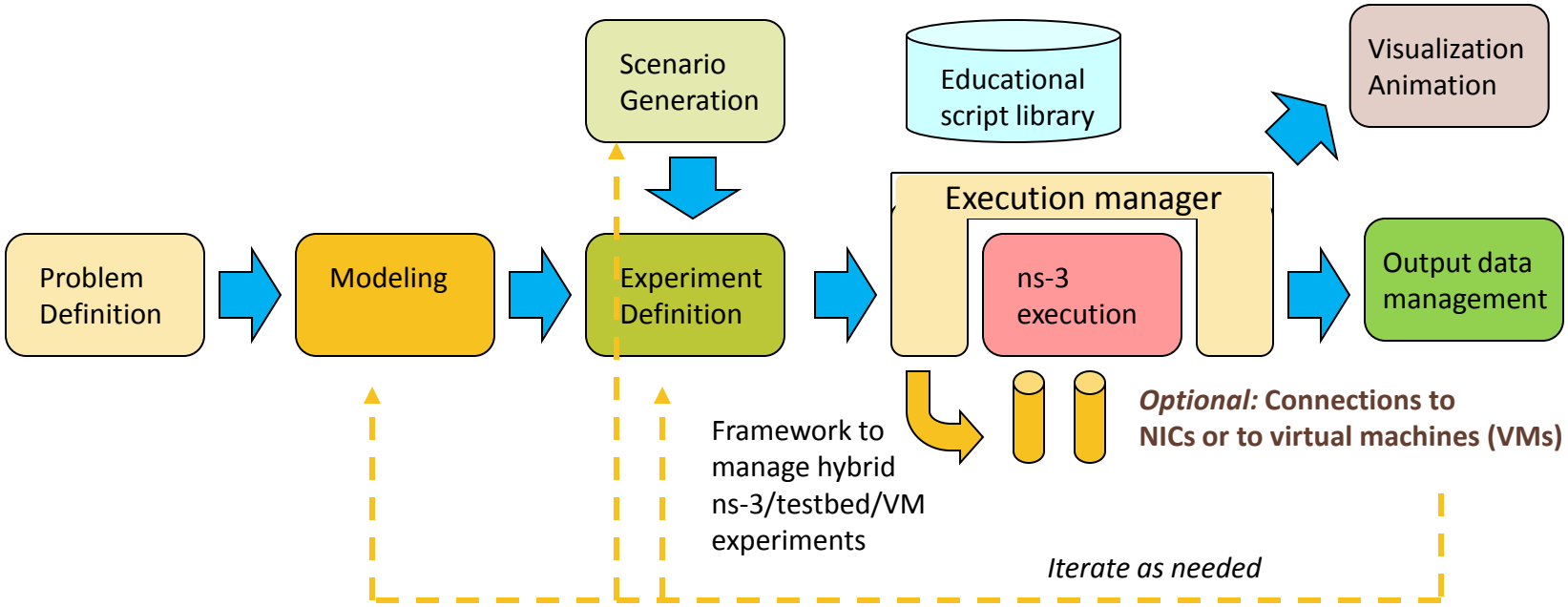
```
Ptr<MyClass> p = CreateObject<MyClass> ();
p->method ();
```

ns-3 Validation

- Can you trust ns-3 simulations?
  - Can you trust any simulation?
  - Onus is on the researcher to verify results
- ns-3 strategies:
  - Open source benefits
  - Validation of models on testbeds
  - Reuse of code
  - Unit tests
  - Event driven validation tests

Frameworks for ns-3

- What do we mean by frameworks?
  - Extensions to ns-3 outside of the core and models
  - Helping users with their workflow



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

Documentation

- In the **`examples/`** and **`samples/`** subdirectories of the source download package
- In the doxygen pages: **`http://nsnam.org/doxygen/`**
- On the project website: **`http://nsnam.org/`**
- A longer, more comprehensive tutorial is online at:
  - **`http://www.nsnam.org/tutorials/simutools08/`**