Introduction to Network Simulator 3

Lalith Suresh

June 19, 2010
Overview of This Presentation

- Network Simulator 2
- Network Simulator 3
- NS-3 Basics
- NS-3 Code Organisation
- NS-3 Node Structure
  - How a node sends a packet
  - How a node receives a packet
  - Components of a node
- Demo/Walkthrough
Network Simulator 2

- The most widely used simulator for networking research.
- Went unmaintained for a long period of time.
- Had multiple issues:
  - Outdated code design.
  - Does not scale well compared to some other existing simulators.
  - Does not involve modern programming practices.
    - Smart Pointers?
    - Design patterns?
  - Tracing system was difficult to use.
Network Simulator 3

- The intended successor to NS-2.
- Written from scratch, and is not an evolved version of NS-2.
- Developed with real world integration in mind:
  - Packets are serialised into actual wire frame packet formats.
  - Packet .pcap traces can be analysed via Wireshark.
  - Optional Real Time scheduler.
  - Network Simulation cradle to use the Linux TCP/IP stack under simulation.
  - POSIX emulation.
NS-3 Basics

- NS-3 is written entirely using C++.
- Heavy use of C++ Templates.
- Use of smart pointers to keep memory management burden off the programmer.
- Heavy use of object aggregation.
- Optionally, you may use Python for scripting.
- Uses waf as it’s build system.
Code Organisation

- Node class
  - NetDevice ABC
  - Address types (IPv4, MAC, etc.)
  - Queues
  - Socket ABC
  - IPv4 ABCs
  - Packet Sockets

- Smart pointers
- Dynamic type system
- Attributes
- Callbacks, Tracing
- Logging
- Random Variables

- helper
  - Routing:
    - olsr, global-routing
  - Internet-stack:
    - (ipv4 impl.)
  - Devices:
    - csma | wifi | ...

- node
  - common
  - simulator

- mobility
  - Events
  - Scheduler
  - Time arithmetic

- Packages
  - Packet Tags
  - Packet Headers
  - Pcap/ascii file writing

- Mobility Models
  - (static, random walk, etc.)
NS-3 Node: Sending a packet

```
Application
  ^    ^                  ^
  v    v                  v
Socket::Send() -> UdpSocketImpl
    ^                       ^
    v                       v
::Send() -> UdpL4Protocol
    ^                   ^
    v                   v
::Send() -> Ipv4L3Protocol
    ^               ^
    v               v
::Send() -> ArpIpv4Interface
    ^                 ^
    v                 v
::Send() -> NetDevice
```

::RouteOutput()

::Lookup()
NS-3 Node: Receiving a packet

```
Application
  ::m_rxCallback -> ForwardUp()

[Tcp/Udp]SocketImpl
  ::ForwardUp()

[Tcp/Udp]L4Protocol
  ::Receive()

Ip6L3Protocol
  ::LocalDeliver()

IPv4RoutingProtocol

Protocol Handlers

m_receiveCallback

NetDevice
```
A node *may* have an Ipv4 stack.

A node *may* have a mobility model.

A node can have any number of NetDevices.

A node can have any number of applications running on it.
Demo/Walkthrough

- Building ns-3.
- Writing a script.
- Using NS_LOG.
Thank You!