NS-3 Advanced Tutorial: Visualization and Data Collection

Tom Henderson (University of Washington and Boeing Research & Technology)
L. Felipe Perrone (Bucknell University)

March 2013
Outline

Getting visualization and raw data from ns-3

• Tracing and packet traces
• Gnuplot and Matplotlib
• Flow Monitor
• PyViz
• NetAnim
• Statistics
• Data Collection Framework
Tracing requirements

• 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
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

```cpp
TracedValue
Config::Connect("/path/to/traced/value", callback1);

TraceSource
Config::Connect("/path/to/trace/source", callback2);

TraceSource unattached
```
NetDevice trace hooks

- Example: CsmaNetDevice

CsmaNetDevice::Send ()

MacTx
MacDrop

queue

MacRx

PhyTxBegin
PhyTxEnd

PhyTxDrop

PhyRxBegin
PhyRxEnd
PhyRxDrop

Sniffer
PromiscSniffer

CsmaNetDevice::TransmitStart ()

CsmaNetDevice::Receive ()

NetDevice::ReceiveCallback

CsmaChannel
Enabling tracing in your code

- examples/tutorial/third.cc

```cpp
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install(p2pNodes);

NodeContainer csmNodes;
csmNodes.Add(p2pNodes.Get(1));
csmNodes.Create(nCsm);

CsmaHelper csm;
csm.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csm.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));

pointToPoint.EnablePcapAll("third");
phy.EnablePcap("third", apDevices.Get(0));
csm.EnablePcap("third", csmDevices.Get(0), true);
```

Device helpers provide common API for enabling pcap traces

Global pcap tracing

Per-device pcap tracing
External pcap tools

wireshark graph analysis

Shawn Ostermann's tcptrace tool
Outline

Getting visualization and raw data from ns-3

• Packet traces
• Gnuplot and Matplotlib
• Flow Monitor
• PyViz
• NetAnim
• Statistics
• Data Collection Framework
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

```cpp
CommandLine cmd;
cmd.Parse (argc, argv);

Gnuplot gnuplot = Gnuplot ("clear-channel.eps");
for (uint32_t i = 0; i < modes.size (); i++)
{
    std::cout << modes[i] << std::endl;
    Gnuplot2dDataset dataset (modes[i]);
}

uint32_t pktsRecvd = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
dataset.Add (rss, pktsRecvd);

gnuplot.AddDataset (dataset);
```

produce a plot file that will generate an EPS figure
one dataset per mode
Add data to dataset
Add dataset to plot
Matplotlib

- Matplotlib or other Python plotting programs can be used
- example: `src/core/examples/sample-rng-plot.py`
Outline

Getting visualization and raw data from ns-3

- Packet traces
- Gnuplot and Matplotlib
- Flow Monitor
- PyViz
- NetAnim
- Statistics
- Data Collection Framework
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

FlowMonitor architecture

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

FlowMonitor statistics

- Statistics gathered
FlowMonitor configuration

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

```cpp
// 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)
    {
        Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
        std::cout << "Flow " << i->first - 2 << " (" << t.sourceAddress << " -> " << t.destinationAddress << ")\n";
        std::cout << "Tx Bytes: " << i->second.txBytes << "\n";
        std::cout << "Rx Bytes: " << i->second.rxBytes << "\n";
        std::cout << "Throughput: " << i->second.rxBytes * 8.0 / 10.0 / 1024 / 1024 << " Mbps\n";
    }
}
```
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 Bytes: 336756
  Throughput: 0.256924 Mbps

Hidden station experiment with RTS/CTS enabled:
Flow 1 (10.0.0.1 -> 10.0.0.2)
  Tx Bytes: 3847500
  Rx Bytes: 396660
  Throughput: 0.233963 Mbps
Flow 2 (10.0.0.3 -> 10.0.0.2)
  Tx Bytes: 3848412
  Rx Bytes: 274740
  Throughput: 0.20961 Mbps
```
Outline

Getting visualization and raw data from ns-3

- Packet traces
- Gnuplot and Matplotlib
- Flow Monitor
- PyViz
- NetAnim
- Statistics
- Data Collection Framework
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 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 (option --enable-sudo not selected)
Build target             :
```

• If program supports CommandLine parsing, pass the option
  ```
  --SimulatorImplementationType=ns3::VisualSimulatorImpl
  ```

• Alternatively, pass the "--vis" option
Getting visualization and raw data from ns-3

- Packet traces
- Gnuplot and Matplotlib
- Flow Monitor
- PyViz
- NetAnim
- Statistics
- Data Collection Framework
NetAnim

• "NetAnim" by George Riley and John Abraham
NetAnim key features

- Animate packets over wired-links and wireless-links
  - 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
NetAnim 3.104 overview

- Forthcoming release
  - More details in packet animation
  - Smoother mobility
  - Plotting the routing path from a source node to a destination IP address
  - Print routing tables at various times
  - Flow monitor output parsing
  - Packet timelines
  - IP/MAC display
  - Change color during animation
  - Designer
Outline

Getting visualization and raw data from ns-3

- Packet traces
- Gnuplot and Matplotlib
- Flow Monitor
- PyViz
- NetAnim
- Statistics
- Data Collection Framework
Statistics

- Statistics module contributed by Joe Kopena early in the project
  - src/stats directory
- Initial implementation of an experiment controller
Statistics module features

- **Metadata**
  - experiment: name of experiment
  - strategy: description of what is being tested
  - runID: allows user to identify the trial

- **Data output in either 'omnetpp' or 'sqlite' format**

- **Provides a basic statistical data calculator**

example "wifi-example-sim" (packet loss vs distance for default wifi settings)
Outline

Getting visualization and raw data from ns-3

• Packet traces
• Gnuplot and Matplotlib
• Flow Monitor
• PyViz
• NetAnim
• Statistics
• Data Collection Framework
Data Collection Framework

• under review for future ns-3 inclusion
• part of the SAFE project development led by Bucknell University
  – integrates with visualization module
  – integrates with steady-state detector
Data Collection Framework

ns-3 data published as trace source

Probe: wrap trace source
- controls to enable/disable
- named within configuration namespace

Collector: data reduction
- examples: averaging, time series, etc.
- can be chained together

Aggregator: marshal data
- gnuplot
- postgresql
- other...

Static method for instrumenting code (Stat::Put() of ns2measure)

Leverages prototype developed by Pavel Boyko and Kirill Andreev
Leverages ns2measure project (CNG at University of Pisa)

NS-3 Consortium Meeting
March 2013
Data Collection Framework example

- `'manet-safe.cc' example in ns-3-dcf repository`
- **Trace source:** `/NodeList/*/ApplicationList/0/$ns3::PacketSink/Rx`

Probe packet sink receptions between time 120-150 seconds

Set periodicity to 0.5 seconds

Plot packet count, total packet byte count (during interval) and mean packet byte count (within interval)
Data Collection Framework

PacketSink

- trace source

Probe

- filter trace source data within time window

Collector

- compute statistics on packet and byte counts

Aggregator

- gnuplot
- postgresql
- SAFE
- other...

Introduce helper to manage configuration complexity
Gnuplot data collection example

- `src/data-collection/examples/manet-safe.cc`

```c
// Configure the plot.
packetPlotHelper.ConfigurePlot("manet-safe-packet-byte-count",
    "MANET SAFE Packet Aggregator",
    "Time (Seconds)",
    "Count",
    "png");

// Add a probe to the gnuplot helper.
packetPlotHelper.AddProbe("ns3::ApplicationPacketProbe",
    "PacketSinkRxProbe",
    "/NodeList/*/ApplicationList/0/ns3::PacketSink/RX");

// Get a pointer to the helper’s probe so that it can be configured.
Ptr<Probe> packetProbe = packetPlotHelper.GetProbe("PacketSinkRxProbe");
packetProbe->SetAttribute("Start", TimeValue(Seconds(120.0)));
packetProbe->SetAttribute("Stop", TimeValue(Seconds(150.0)));

// Add a collector to the gnuplot helper.
packetPlotHelper.AddCollector("ns3::BasicStatsCollector",
    "PacketSinkRxCollector",
    "PacketSinkRxProbe",
    "OutputBytes");

// Get a pointer to the helper’s collector so that it can be configured.
Ptr<Collector> packetCollector = packetPlotHelper.GetCollector("PacketSinkRxCollector");
packetCollector->SetPeriodic(Seconds(0.5));

// Get a pointer to the helper’s aggregator so that it can be configured.
Ptr<GnuplotAggregator> packetAggregator = packetPlotHelper.GetAggregator();
packetAggregator->Set2dDatasetDefaultStyle(Gnuplot2dDataset::POINTS);
```
Gnuplot data collection example (2)

- src/data-collection/examples/manet-safe.cc

```cpp
// Add some datasets to the plot. Note that the dataset context
// strings, which are the third arguments in these function calls,
// must be unique
packetPlotHelper.Add2dDataset
    ("PacketSinkRxCollector",
    "SampleCount",
    "PacketSinkRxCollector/SampleCount",
    "Packet Count");
packetPlotHelper.Add2dDataset
    ("PacketSinkRxCollector",
    "SampleSum",
    "PacketSinkRxCollector/SampleSum",
    "Total Packet Byte Count");
packetPlotHelper.Add2dDataset
    ("PacketSinkRxCollector",
    "SampleMean",
    "PacketSinkRxCollector/SampleMean",
    "Mean Packet Byte Count");

// Set this dataset's sytyle.
packetAggregator->Set2dDatasetStyle
    ("PacketSinkRxCollector/SampleCount",
    GnuPlot2dDataset::LINES_POINTS);
```
Under construction:
steady-state collector

- Built upon steady-state detector classes
- Receives samples
- Applies steady-state detection algorithm
- One pass-through trace source (all samples)
- One filtered trace source (post-transient sample)
- MSER-5 and possibly other methods
Under construction: in-browser visualization

Selection buttons for metrics collected

Time series support: nearly completed
Plot is built interactively, through browser, and converted to static file (PDF)
Next priority for in-browser visualization

“Estimation” support: in early design

Selection buttons for data series

Confidence intervals

Figure source: Broch et al. “A Performance Comparison of Multi-Hop Wireless Ad Hoc Routing Protocols”
Later priority for in-browser visualization

“Distribution” support: in early discussion

Selection buttons for data series

Figure source: Broch et al. “A Performance Comparison of Multi-Hop Wireless Ad Hoc Routing Protocols”