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):

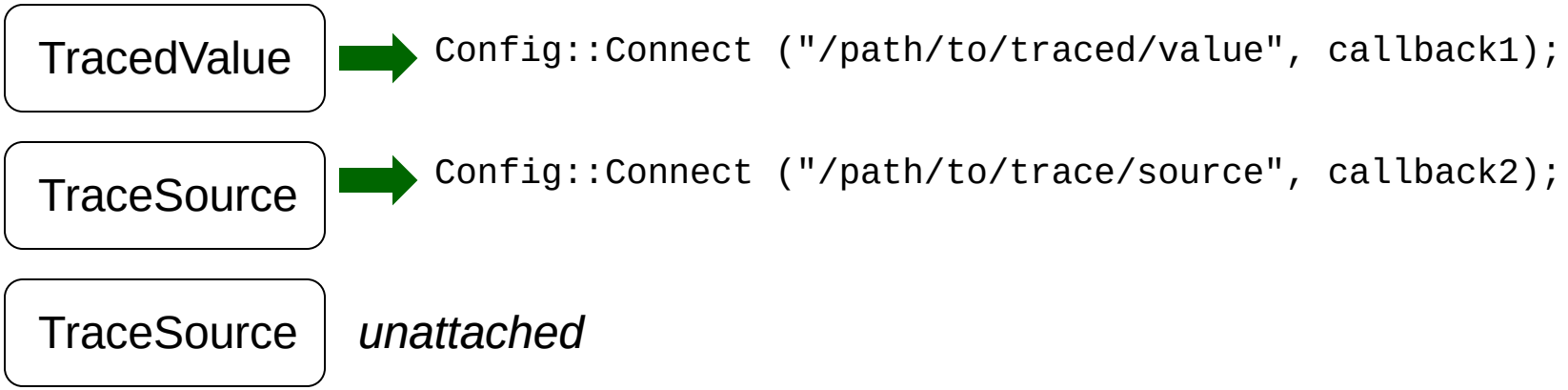
```
+ 1.84375 0 2 cbr 210 ------- 0 0.0 3.1 225 610
- 1.84375 0 2 cbr 210 ------- 0 0.0 3.1 225 610
r 1.84471 2 1 cbr 210 ------- 1 3.0 1.0 195 600
r 1.84566 2 0 ack 40 ------- 2 3.2 0.1 82 602
+ 1.84566 0 2 tcp 1000 ------- 2 0.1 3.2 102 611
```

Problem: Tracing needs vary widely

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

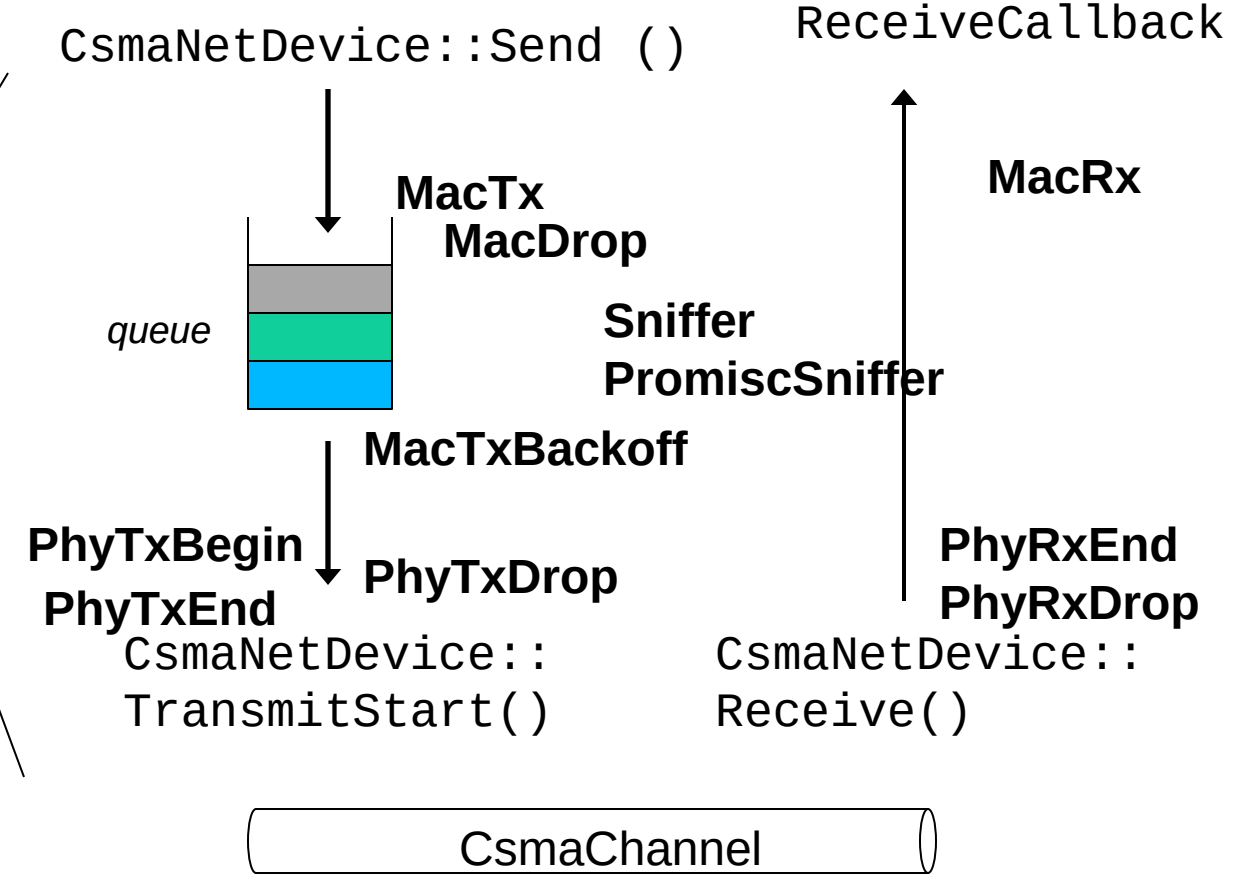
Tracing in ns-3

- 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



NetDevice trace hooks

- Example: CsmaNetDevice



Enabling tracing in your code

- examples/tutorial/third.cc

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

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

NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);

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

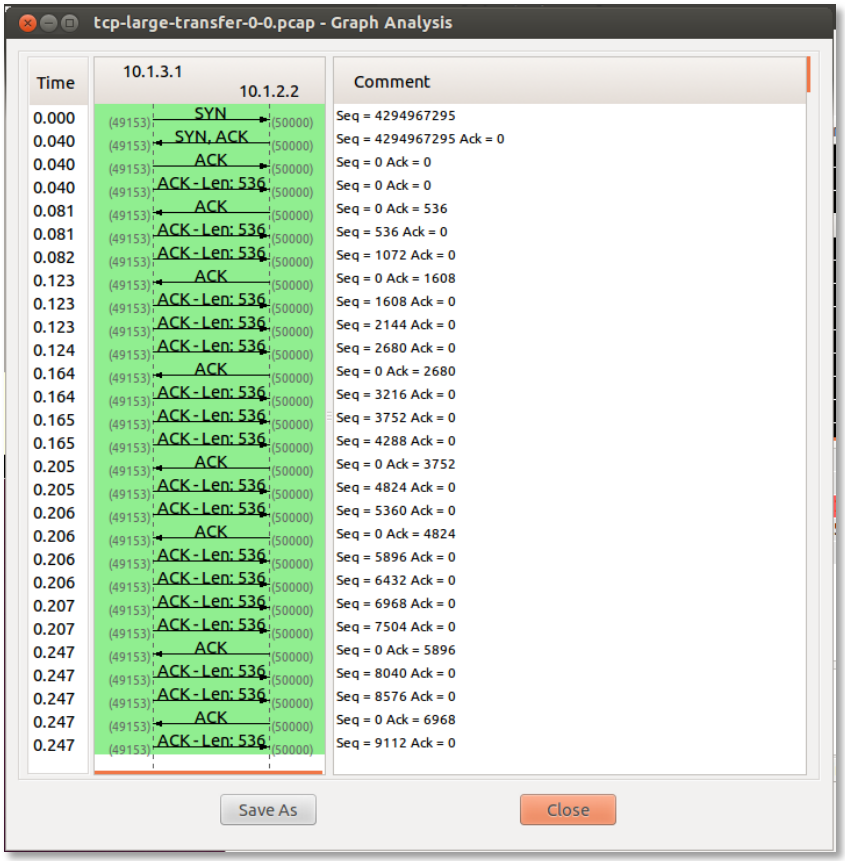
Device helpers provide common API for enabling pcap traces

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

Global pcap tracing

Per-device pcap tracing

External pcap tools



wireshark graph analysis

```
1 arg remaining, starting with 'tcp-large-transfer-1-0.pcap'
Ostermann's tcptrace -- version 6.6.7 -- Thu Nov  4, 2004

5604 packets seen, 5604 TCP packets traced
elapsed wallclock time: 0:00:10.388284, 539 pkts/sec analyzed
trace file elapsed time: 0:00:02.173805
TCP connection info:
1 TCP connection traced:
TCP connection 1:
        host a:         10.1.3.1:49153
        host b:         10.1.2.2:50000
        complete conn: yes
        first packet:   Wed Dec 31 16:00:00.010033 1969
        last packet:    Wed Dec 31 16:00:02.183839 1969
        elapsed time:   0:00:02.173805
        total packets: 5604
        filename:       tcp-large-transfer-1-0.pcap
   a->b:                                b->a:
     total packets:          3735           total packets:          1869
     ack pkts sent:          3734           ack pkts sent:          1869
     pure acks sent:            2           pure acks sent:         1867
     sack pkts sent:            0           sack pkts sent:            0
     dsack pkts sent:           0           dsack pkts sent:           0
     max sack blks/ack:         0           max sack blks/ack:         0
     unique bytes sent:   2000000           unique bytes sent:         0
     actual data pkts:       3732           actual data pkts:          0
     actual data bytes:   2000000           actual data bytes:         0
     rexmt data pkts:           0           rexmt data pkts:           0
     rexmt data bytes:          0           rexmt data bytes:          0
     zwnd probe pkts:           0           zwnd probe pkts:           0
     zwnd probe bytes:          0           zwnd probe bytes:          0
     outoforder pkts:           0           outoforder pkts:           0
     pushed data pkts:          0           pushed data pkts:          0
     SYN/FIN pkts sent:       1/1           SYN/FIN pkts sent:       1/1
```

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`

```
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]);
```

produce a plot file that will generate an EPS figure

one dataset per mode

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

gnuplot.AddDataset (dataset);
```

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`

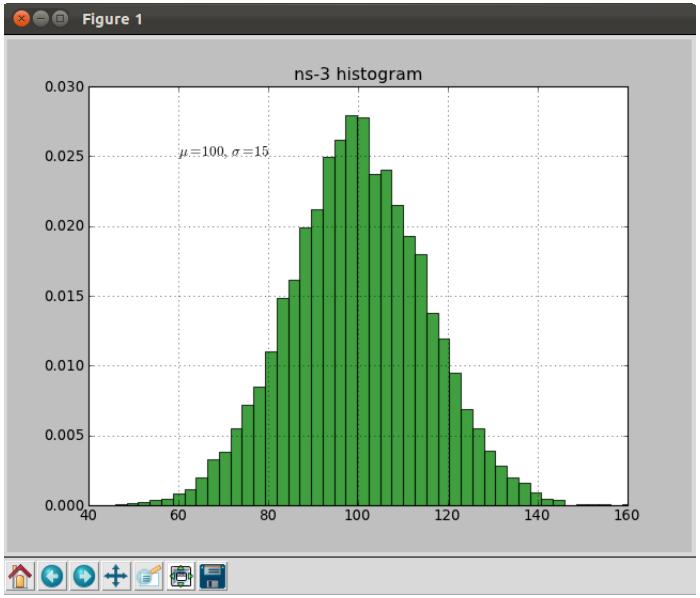
```
# Demonstrate use of ns-3 as a random number generator integrated with
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial

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()
```



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

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

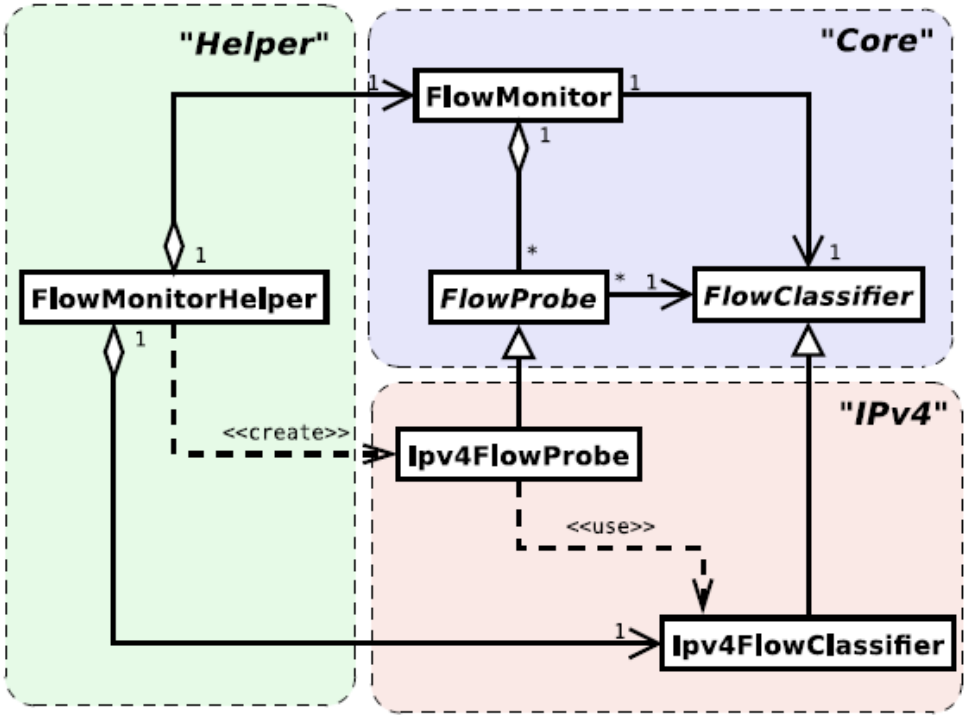
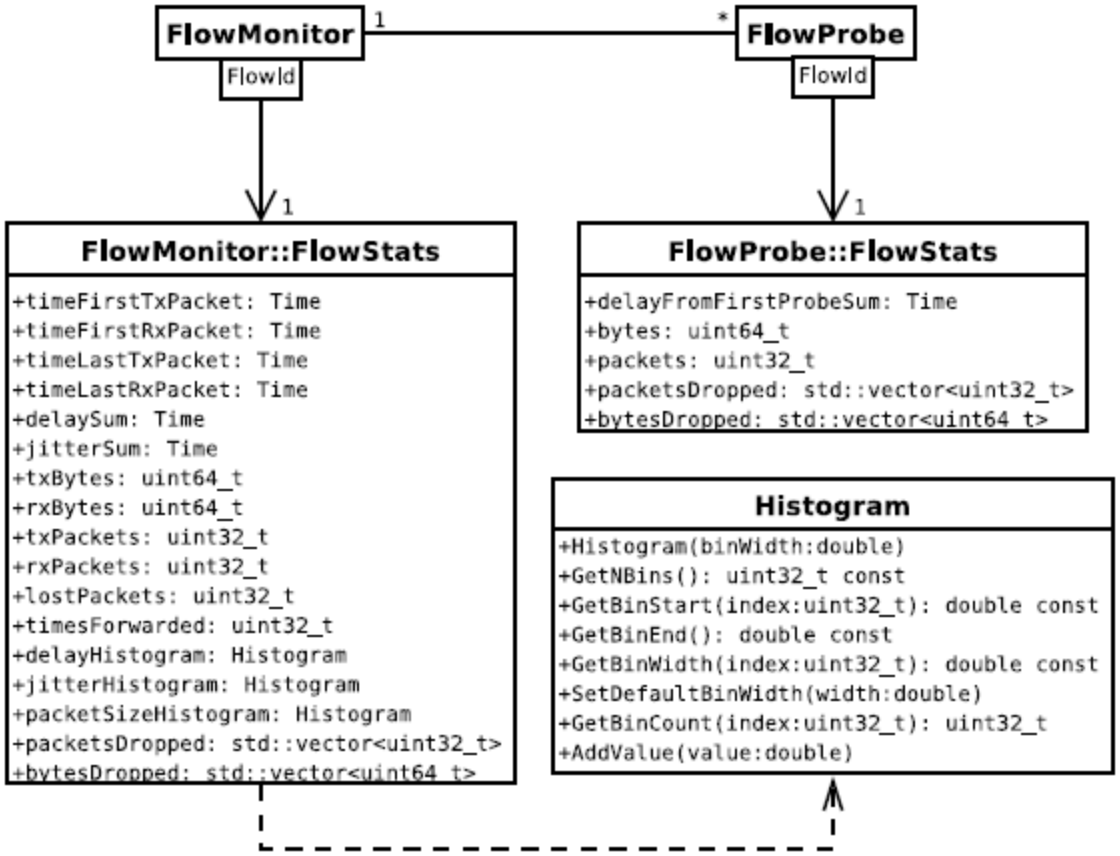


Figure credit: G. Carneiro, P. Fortuna, M. Ricardo, "FlowMonitor-- a network monitoring framework for the Network Simulator ns-3," Proceedings of NSTools 2009.

FlowMonitor statistics

- Statistics gathered



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)
      {
        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:   306660
  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

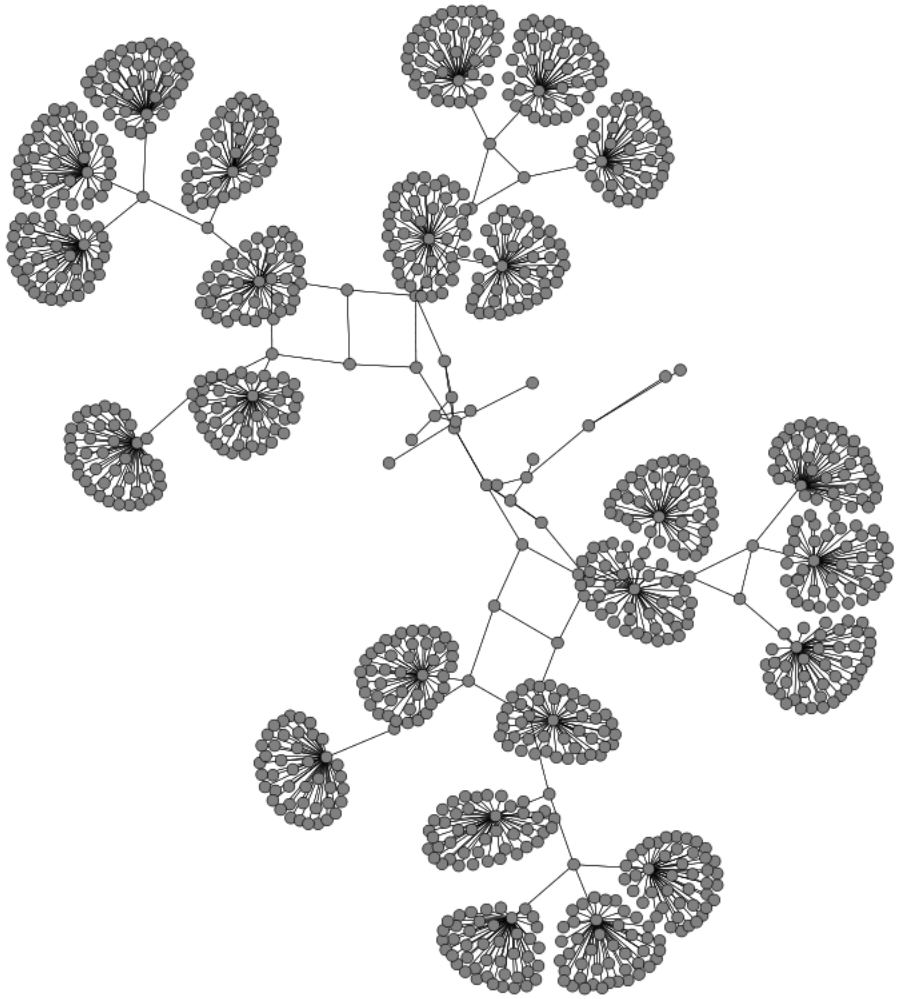
Getting visualization and raw data from ns-3

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

PyViz overview

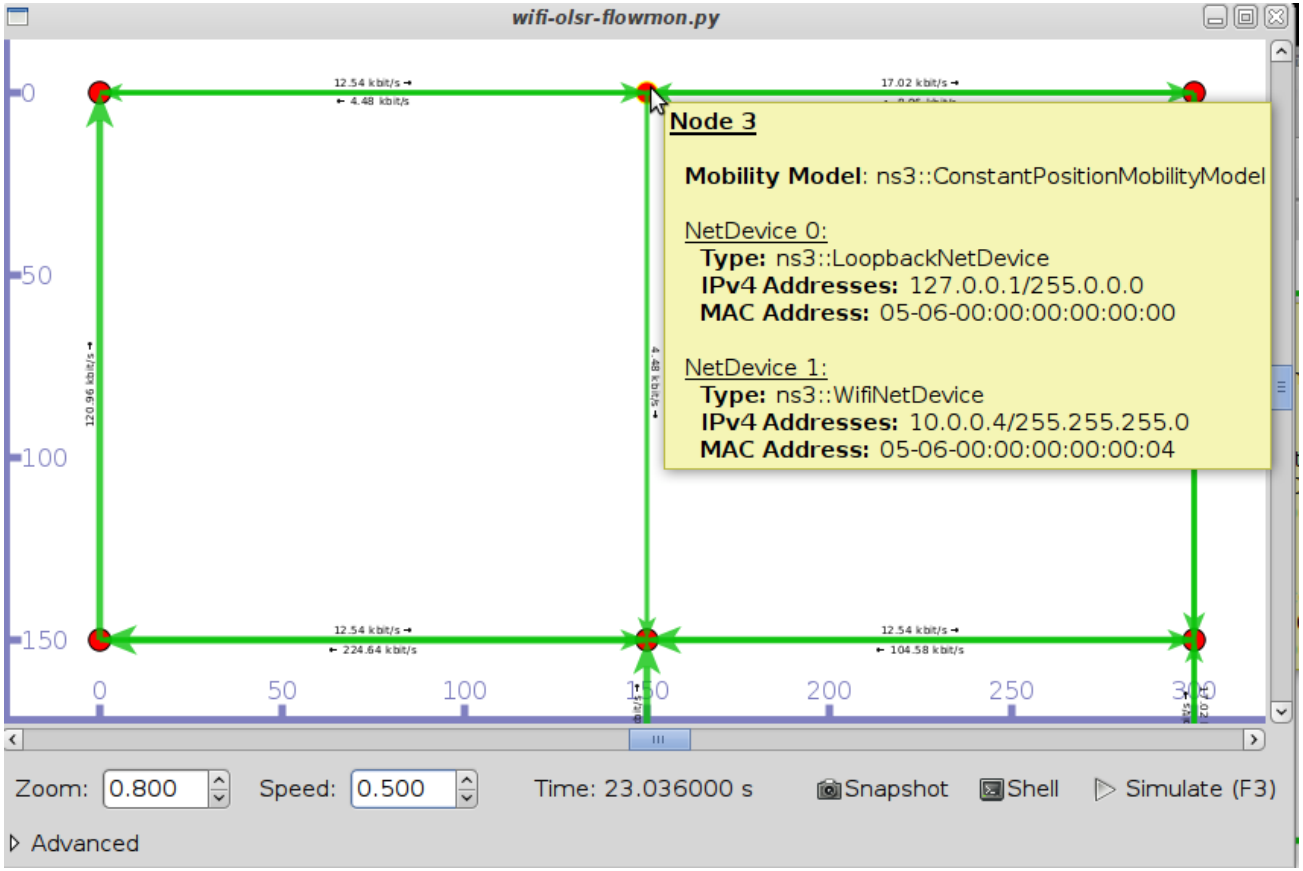
- Developed by Gustavo Carneiro
- Live simulation visualizer (no trace files)
- Useful for debugging
  - mobility model behavior
  - where are packets being dropped?
- Built-in interactive Python console to debug the state of running objects
- Works with Python and C++ programs

Pyviz 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 (option --enable-sudo not selected)
```

- If program supports CommandLine parsing, pass the option

  `--SimulatorImplementationType=ns3::VisualSimulatorImpl`

- Alternatively, pass the "--vis" option

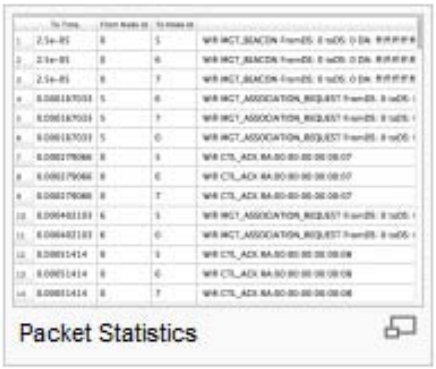
Outline

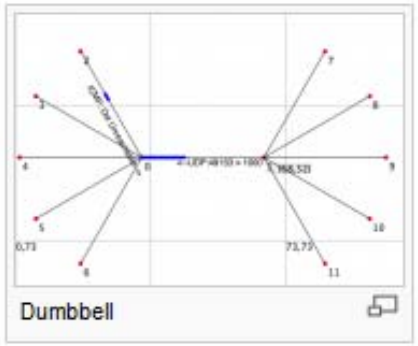
Getting visualization and raw data from ns-3

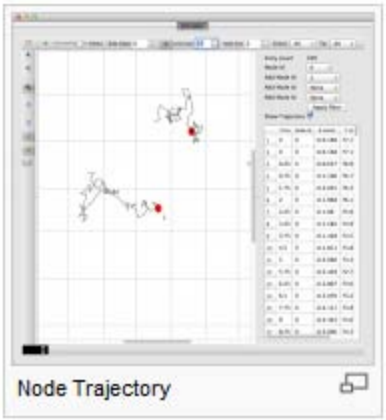
- Packet traces
- Gnuplot and Matplotlib
- Flow Monitor
- PyViz
- NetAnim
- Statistics
- Data Collection Framework

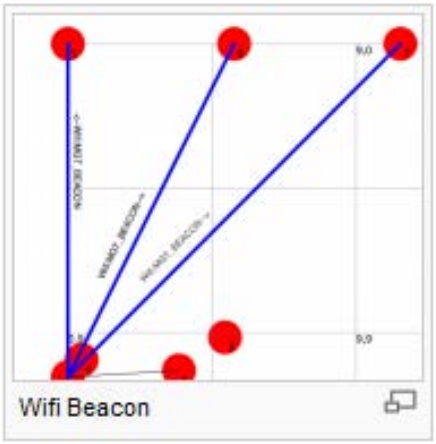
NetAnim

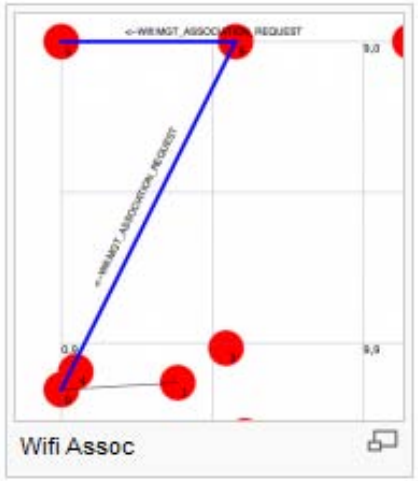
- "NetAnim" by George Riley and John Abraham

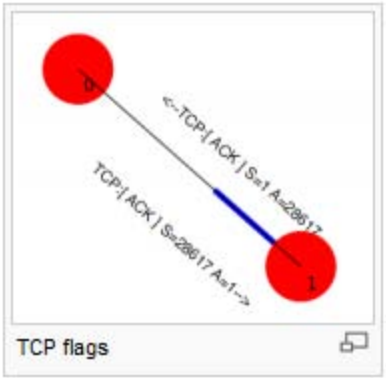

Packet Statistics


Dumbbell


Node Trajectory


Wifi Beacon


Wifi Assoc


TCP flags

NetAnim key features

- Animate packets over wired-links and wireless-links
  - limited support for LTE traces
- Packet timeline with regex filter on packet meta-data.
- 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

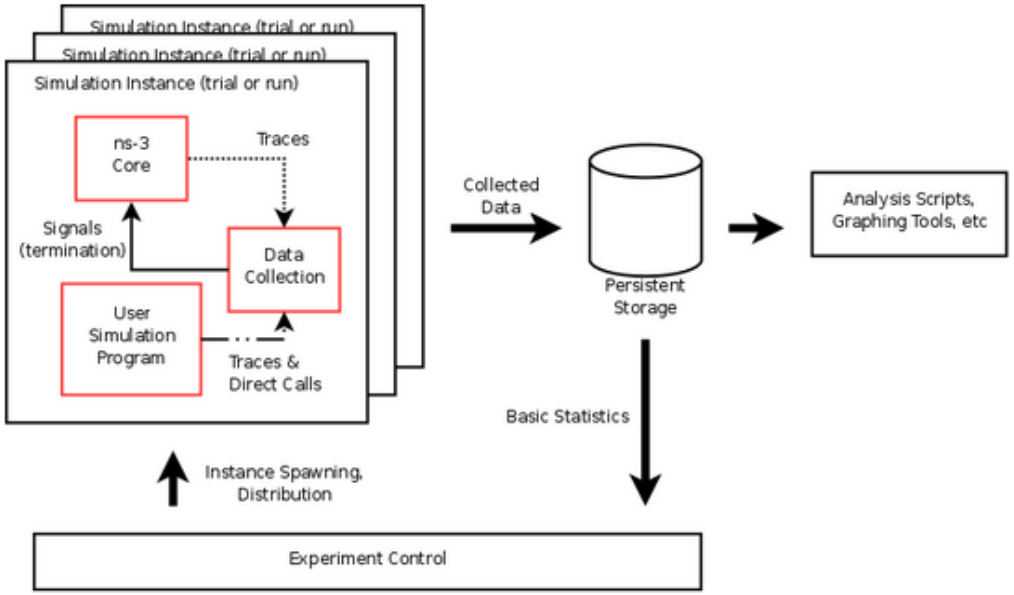
Outline

Getting visualization and raw data from ns-3

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

Statistics

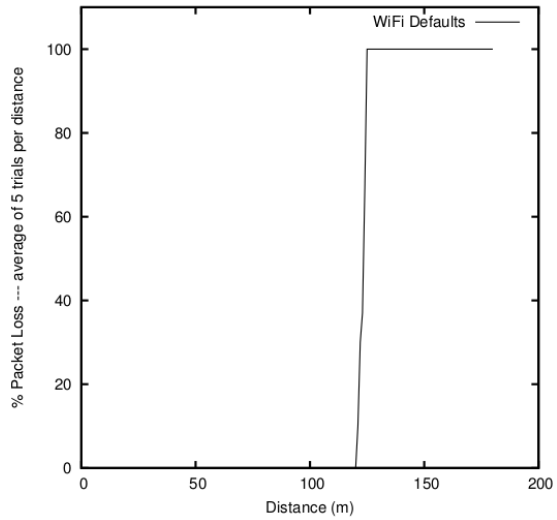
- Statistics module contributed by Joe Kopena early in the project
  - `src/stats` directory
- Initial implementation of an experiment controller



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

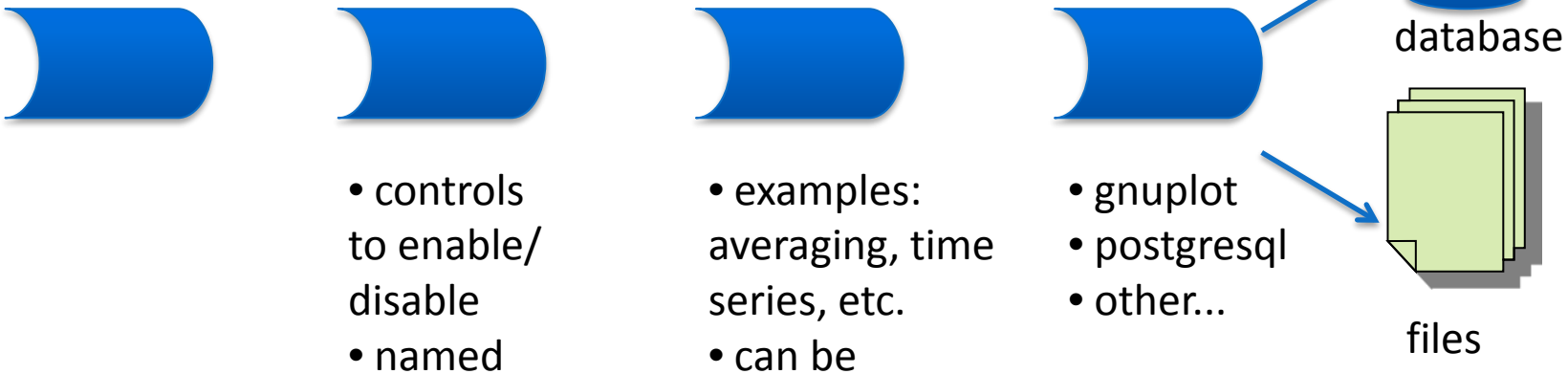
Data Collection Framework

ns-3 data published as trace source

Probe: wrap trace source

Collector: data reduction

Aggregator: marshal data



- controls to enable/ disable
- named within configuration namespace

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

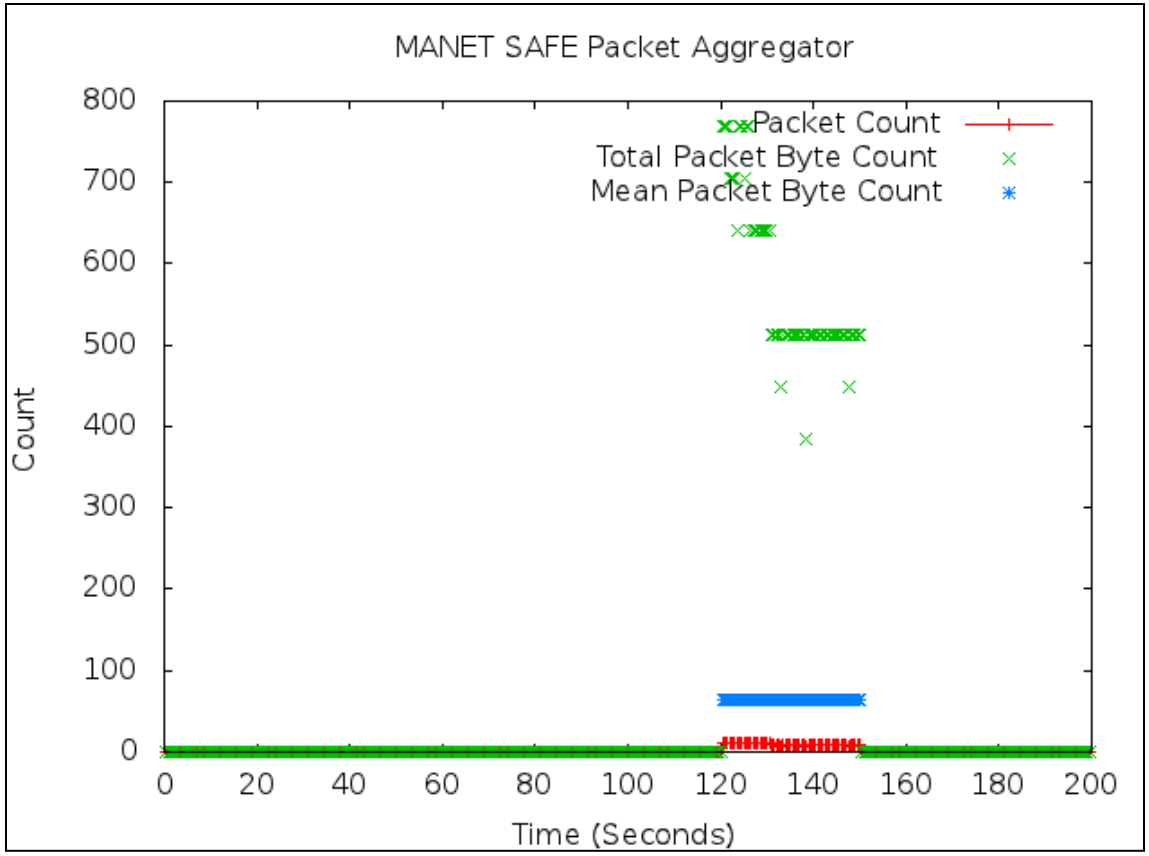
- gnuplot
- postgresql
- other...

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)

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

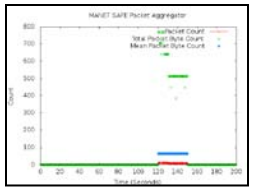
- filter trace source data within time window

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`

```
// 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`

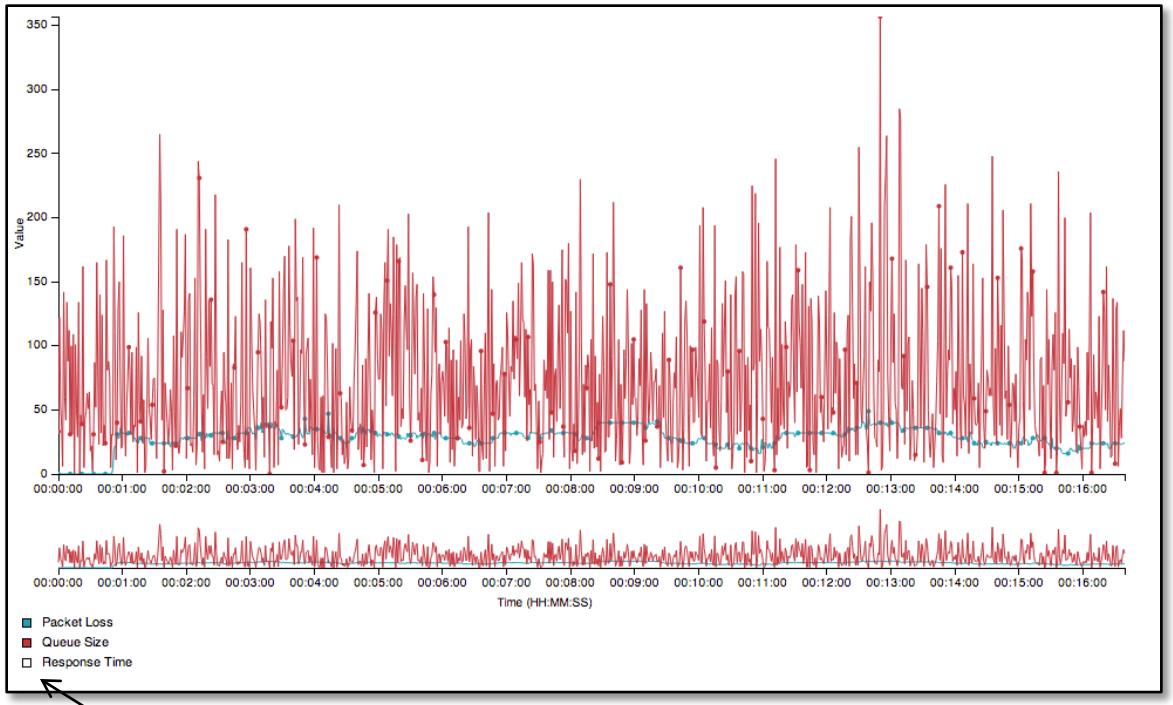
```
// Add some datasets to the plot.  Note that the dataset context
// strings, which are the third arguments in these function calls,
// must be unique
packetPlotHelper.Add2dDataset ("PacketSinkRxCollector",
                               "SampleCount",
                               "PacketSinkRxCollector/SampleCount",
                               "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

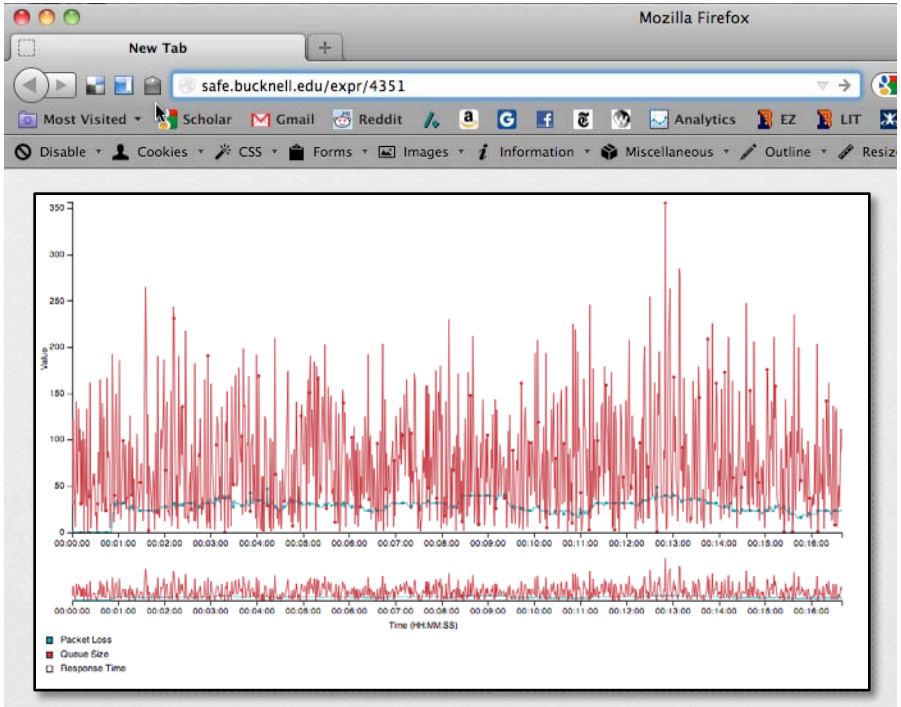
Under construction: in-browser visualization

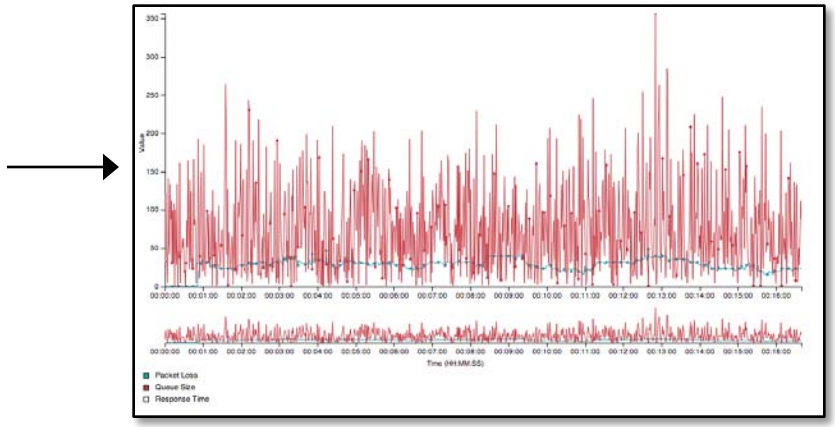


Time series support: nearly completed

Selection buttons for metrics collected

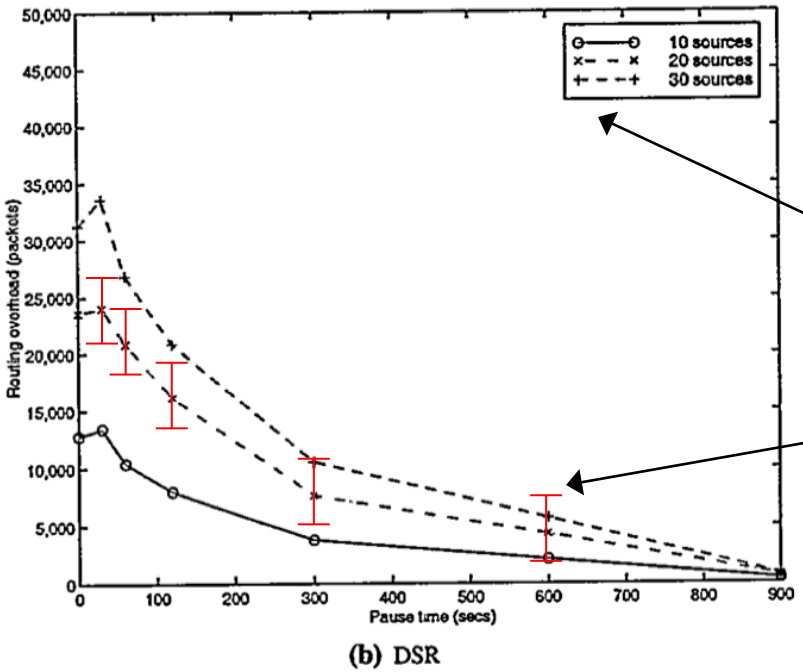
Under construction: from browser to file





Plot is built interactively, through browser, and converted to static file (PDF)

Next priority for in-browser visualization



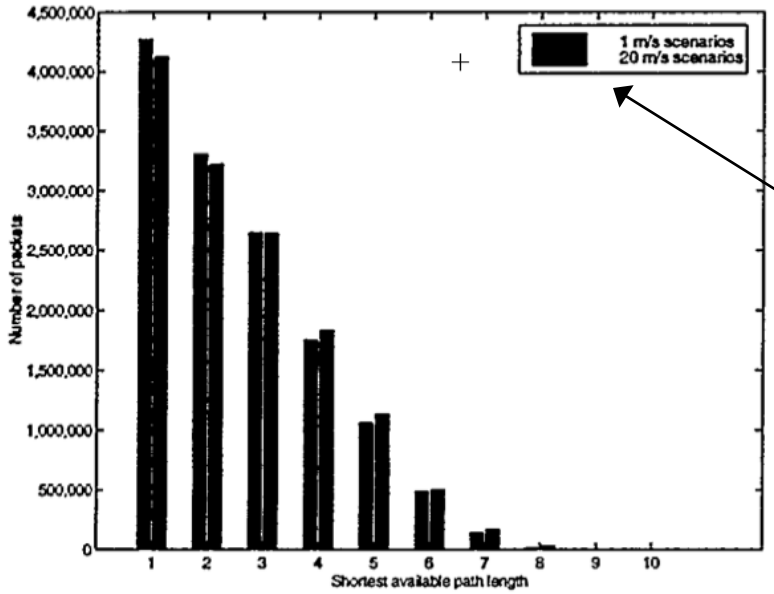
"Estimation" support: in early design

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"