Data collection framework code review

- This code review is for the March developers meeting
- Would like feedback/concurrence on moving towards merging some of this code
- Data collection framework is being worked as part of the SAFE project, which is also working on automation, transient detection, and other support software for ns-3
  - See: http://www.eg.bucknell.edu/~perrone/research/

ns-3 NETWORK SIMULATOR

High level objectives of DCF

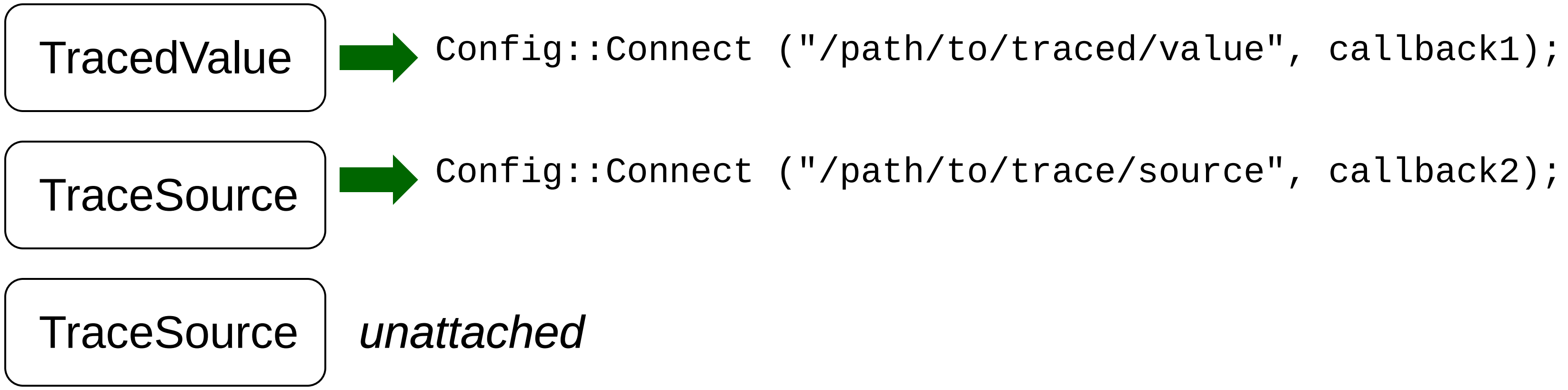
- Help users get data out of the simulator and into plots, files and databases
- Provide statistical support for transforming data into means, error bars, etc.
- Feed into other (future) elements of SAFE such as transient detectors
- Data provenance: record where the data came from, and make it reproducible

High level design

- ns-3 trace sources (traced values, traced callbacks) provide mechanism to export data
  - build on top of trace source mechanism
  - provide ways for users to insert their own custom trace source without too much hassle

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



```
Config::Connect ("/path/to/traced/value", callback1);
Config::Connect ("/path/to/trace/source", callback2);
```

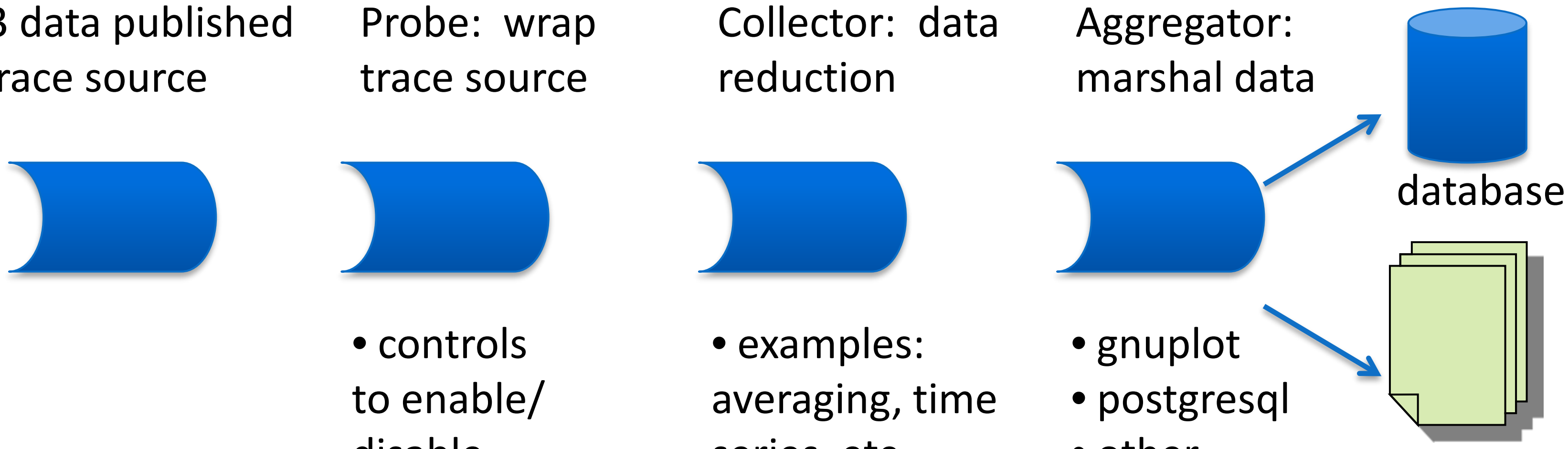
Data Collection Framework

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

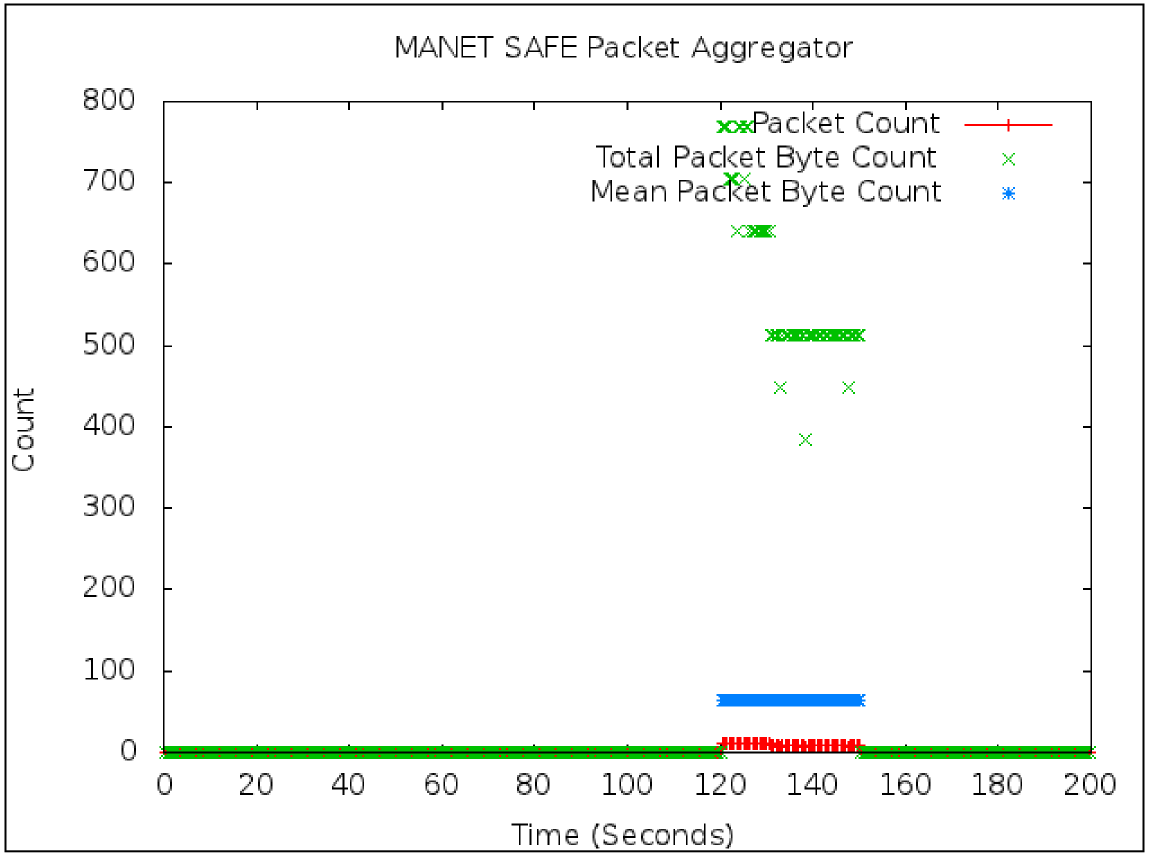
- gnuplot
- postgresql
- other...

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

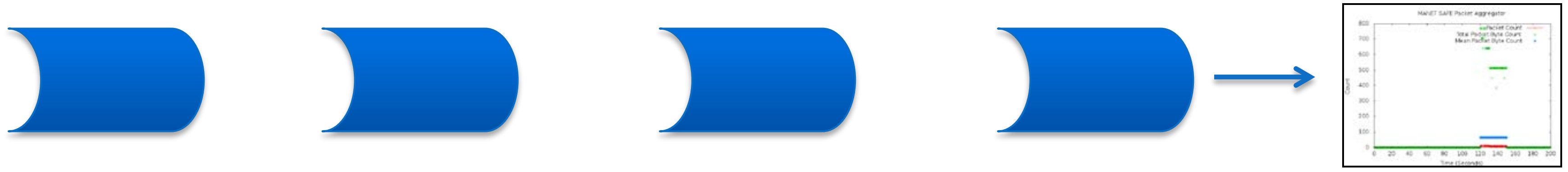
- filter trace source data within time window

Collector

- compute statistics on packet and byte counts

Aggregator

- gnuplot
- postgresql
- other...



Introduce helper to manage configuration complexity

ns-3 NETWORK SIMULATOR

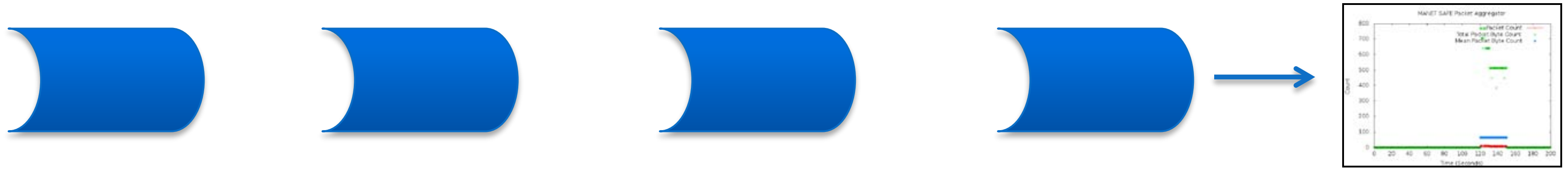
Current scope for this code review

PacketSink trace source

Probe

Collector

Aggregator



we've added a few new trace sources in our SAFE repository, for experimentation purposes

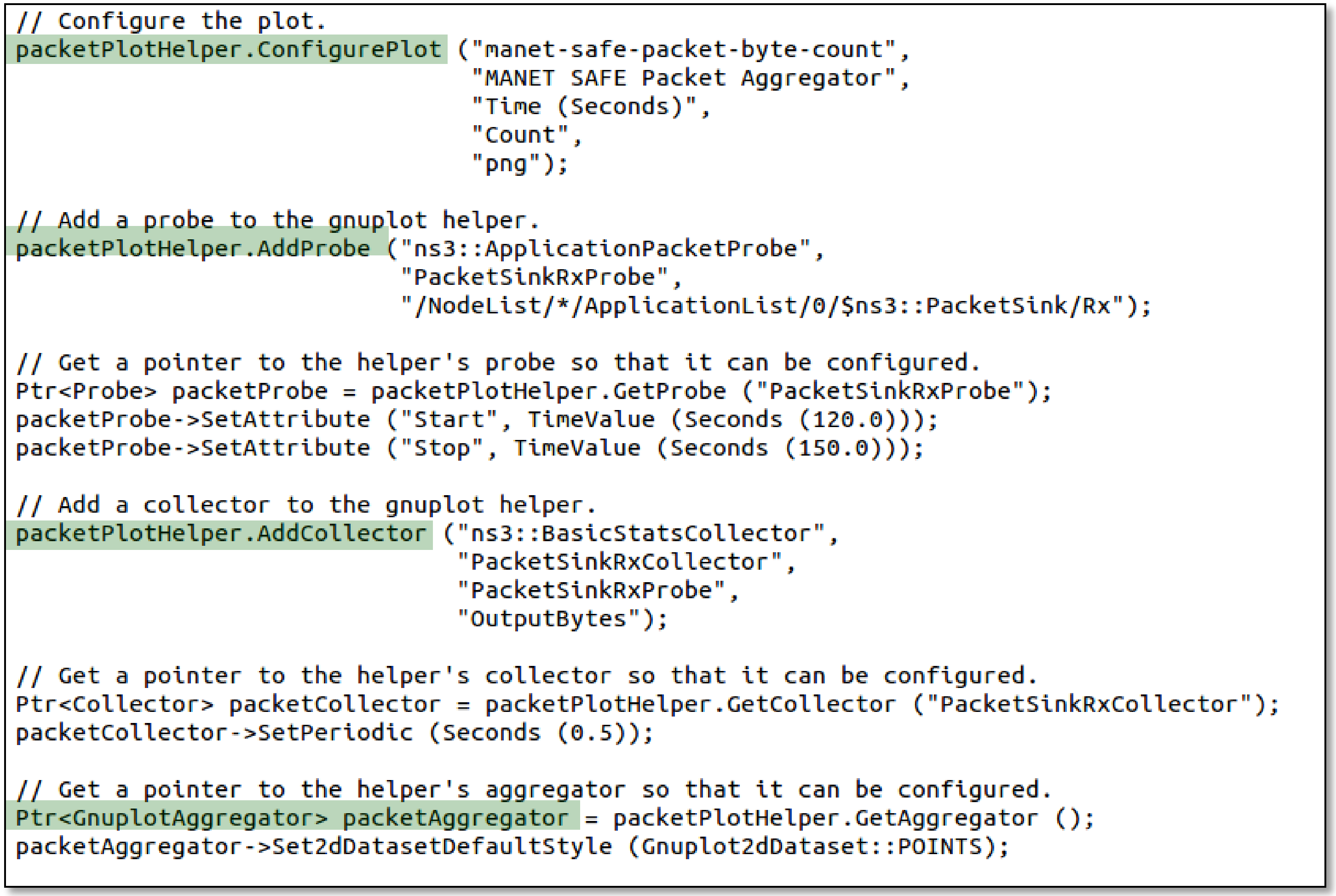
a few Probe classes for the dominant data types (Uinteger variants, Double, Packet, Ipv4Packet)

we only have one statistical collector which wraps the stats basic-data-calculators.h file

we have one aggregator implemented (gnuplot) and the helper for it; other aggregators (file, database) are planned

Gnuplot data collection example

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

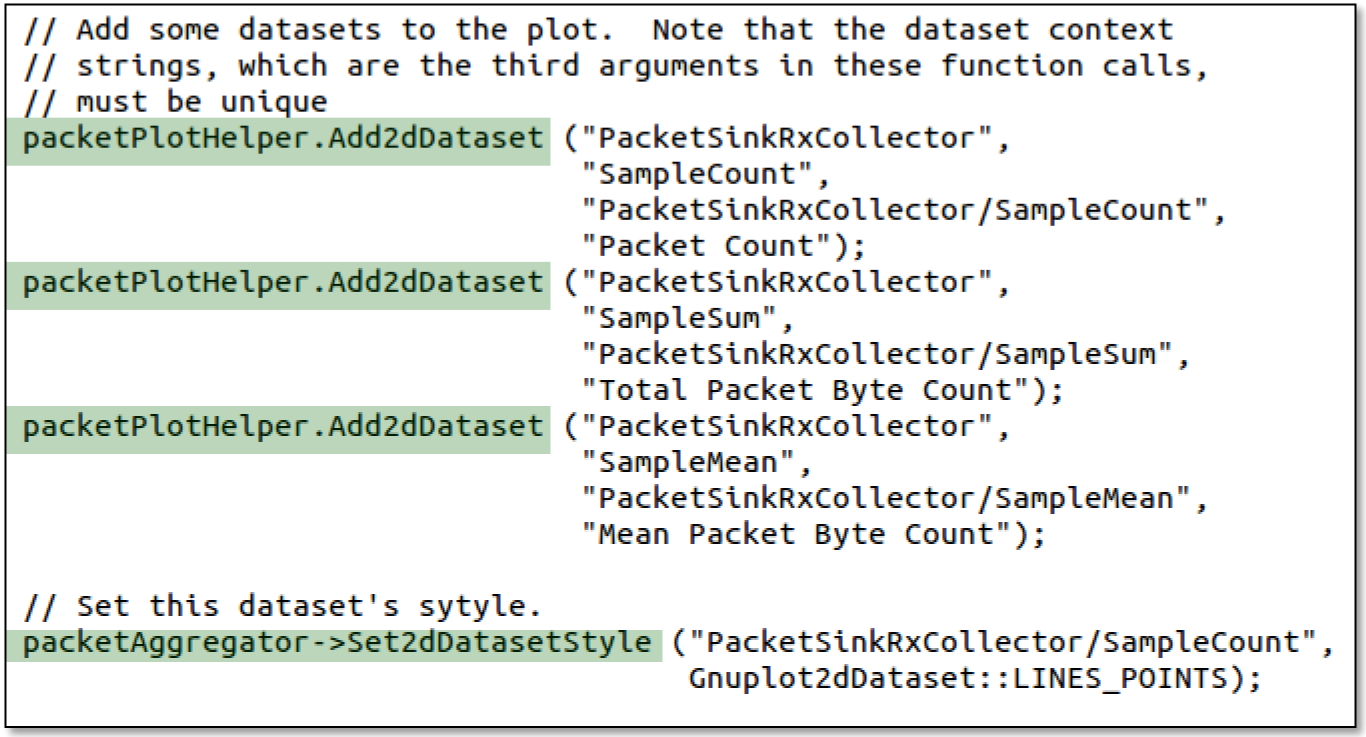


```
// Configure the plot.
packetPlotHelper.ConfigurePlot ("manet-safe-packet-byte-count",
                                "MANET SAFE Packet Aggregator",
                                "Time (Seconds)",
                                "Count",
                                "png");

// 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);
```

NETWORK SIMULATOR

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);
```

Issues (1)

- Probe/collector interface
  - Mutual support of 1) high-level API that avoids callback notation, 2) possibly many probe trace source types, and 3) want to implement collectors without code duplication
  - This has proven to be difficult, so we are falling back (for now) to make it that all probes export a double typed value

Walkthrough the example to illustrate this

ns-3 NETWORK SIMULATOR

Issues (2)

- Synchronization of data from multiple collectors into aggregators
- Asynchronous API of upstream objects "publishing" downstream is not aligned with need for collectors to sometimes pull (poll) for most current data

Walkthrough the example to illustrate this

Issues (3)

- src/data-collection should probably be added into the module hierarchy at a low-level (e.g. above 'network') so that other modules can define their own probes; is this acceptable?

Desired next steps

- Review/confirm general approach to the problem
- Resolve open issues and provide currently scoped code for a merge review