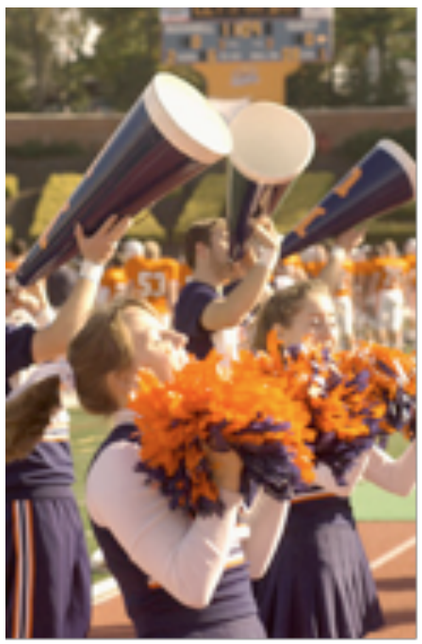
The Design of an Output Data Collection Framework for ns-3

L. Felipe Perrone
Vinícius D. Felizardo

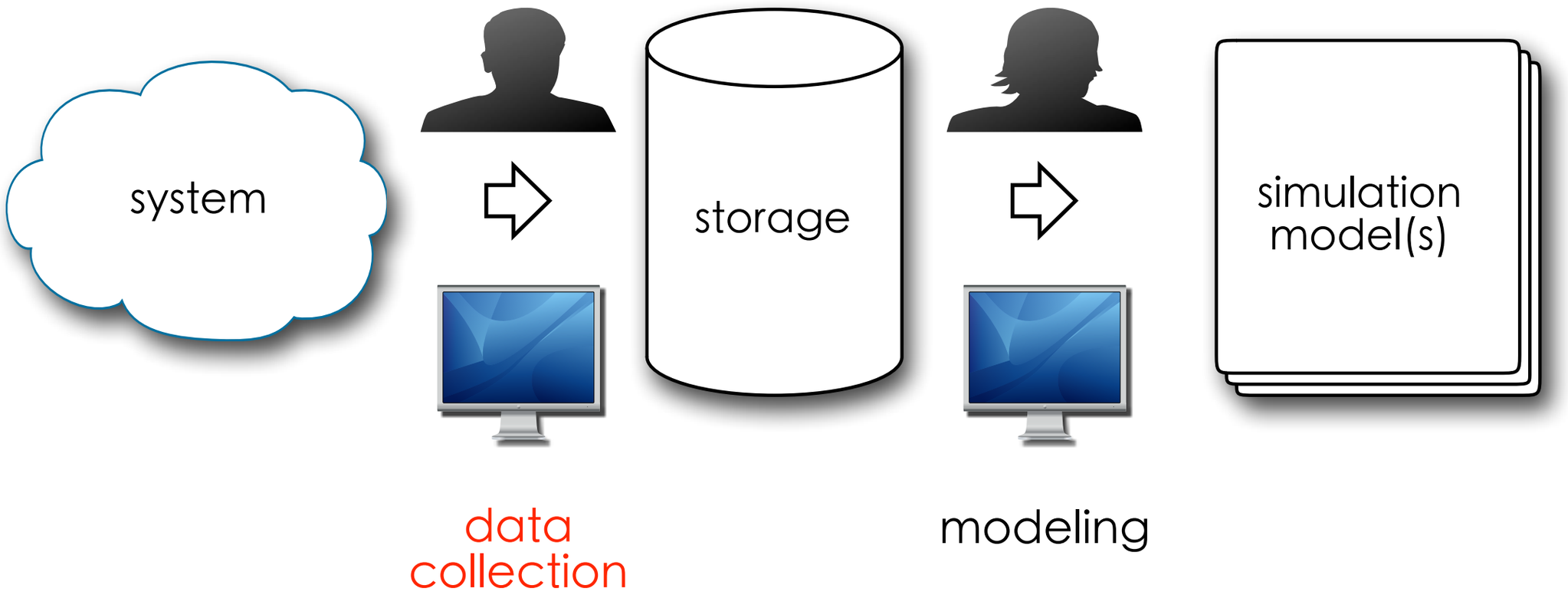
Dept. of Computer Science
Bucknell University, PA, U.S.A.

Thomas R. Henderson (Boeing/UW)
Mitchell J. Watrous

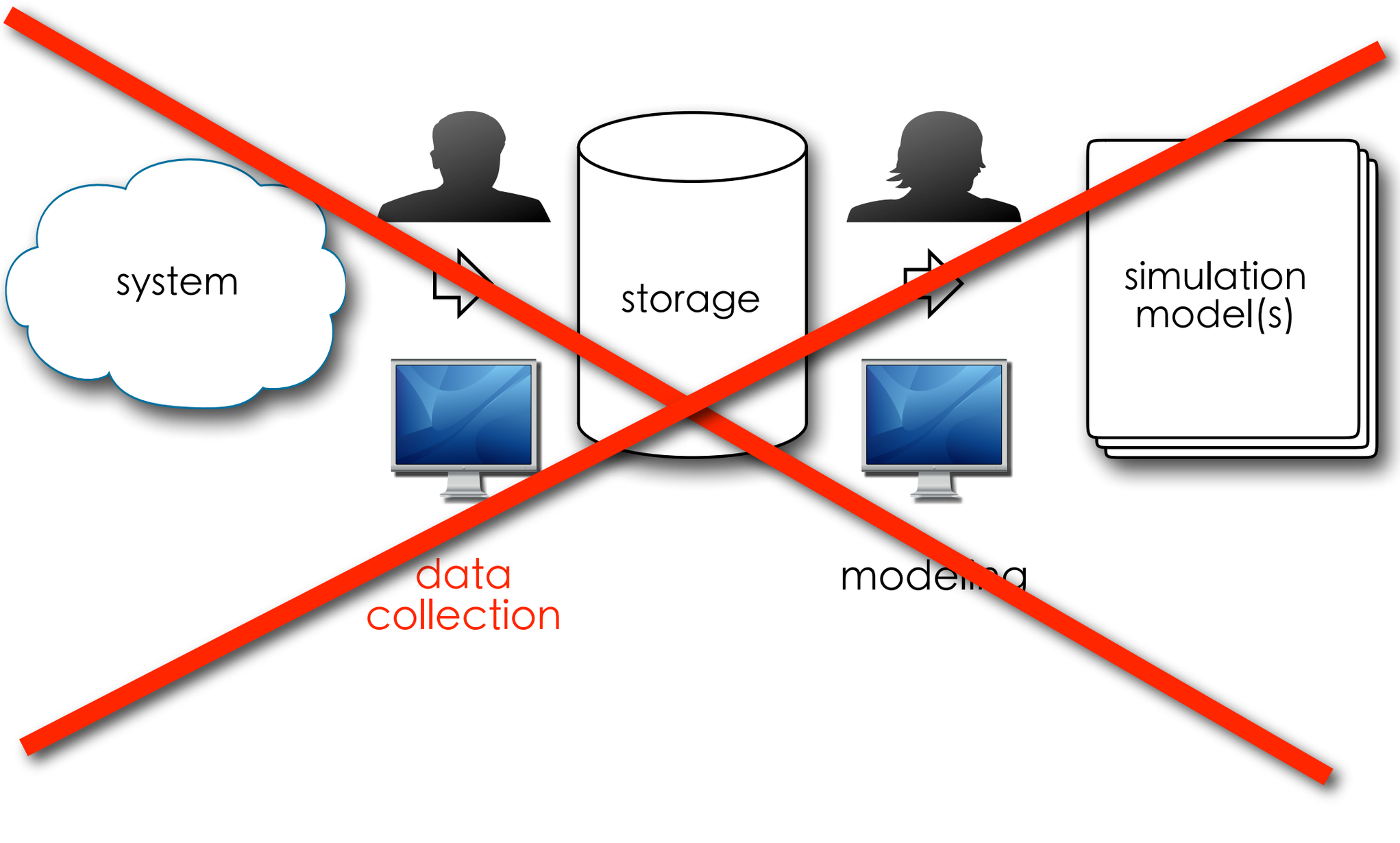
Dept. of Electrical Engineering
University of Washington,
Seattle, WA, U.S.A.



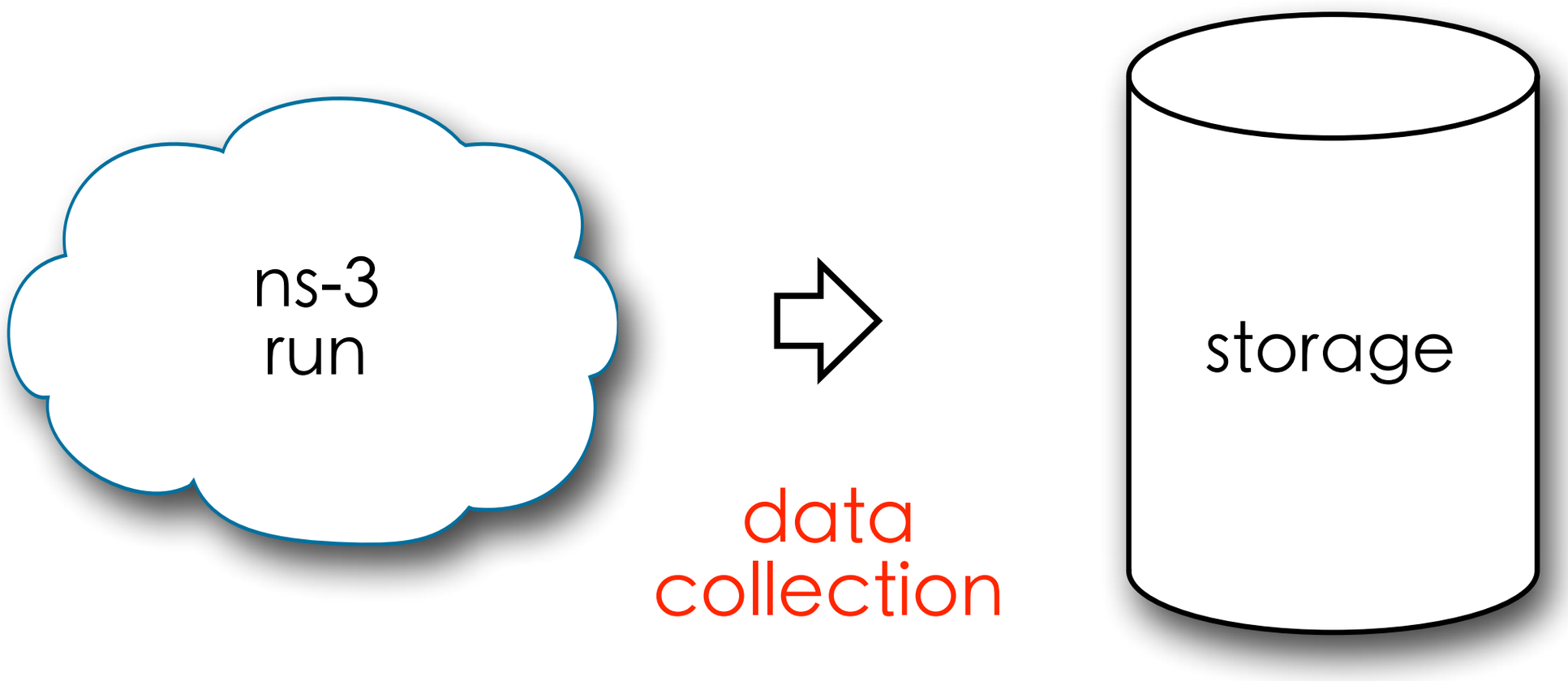

Data Collection



Data Collection



Data Collection



How can you do it?

```
printf(  “%lf\n”  ,  m_double );
```

How can you do it?

printf(“%lf\n” , m_double);

- ad hoc, error prone
- underinstrument, overinstrument
- burden on experimenter
- not easily controllable
- have to parse output to extract data of interest

How can you do it with ns-3?

```
TracedCallback<double> my_double;

TypeId
MyModel::GetTypeId(void) {
  static TypeId = TypeId(  “ns3::MyModel”  )
    .SetParent<Object> ()
    .AddTraceSource(  “MySource”  ,
      “Some comments...”  ,
      MakeTraceSourceAccessor (&m_double)
      ...
}
```

How can you do it with ns-3?

... and then you connect to a trace sink...

```
void DoubleTrace(double old, double new) {
   ...
}
```

```
Ptr<MyModel> myObject =
CreateObject<MyModel> ();

myObject->TraceConnectWithoutContext
("MyDouble", MakeCallback(&DoubleTrace));
```

How can you do it with ns-3?

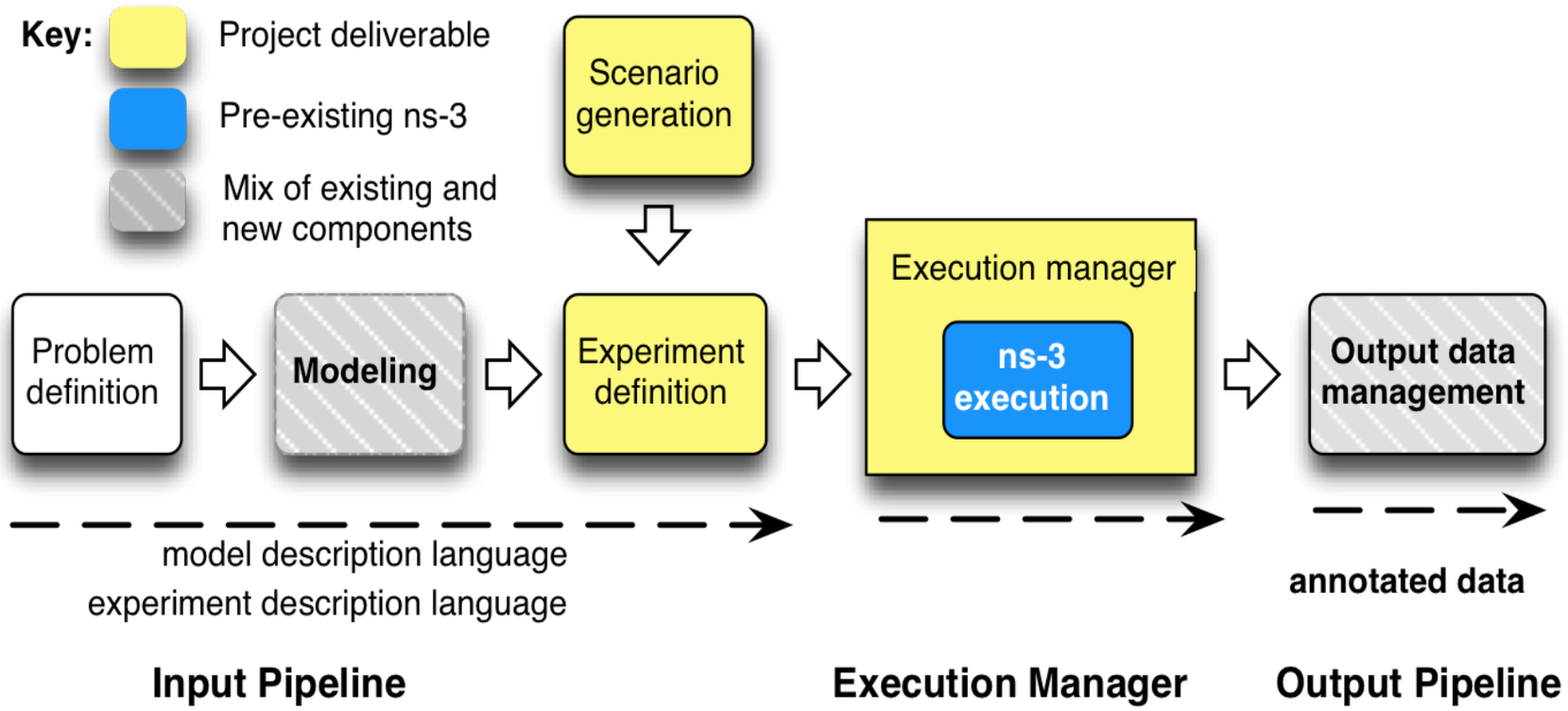
- boiler plate code overhead
- model author defines trace sources
- model user selects trace sources of interest
- source and sink connected by name, or "context" string
- one trace source can map to multiple trace sinks
- have to match signature of source and sink

How can you do it better with ns-3?

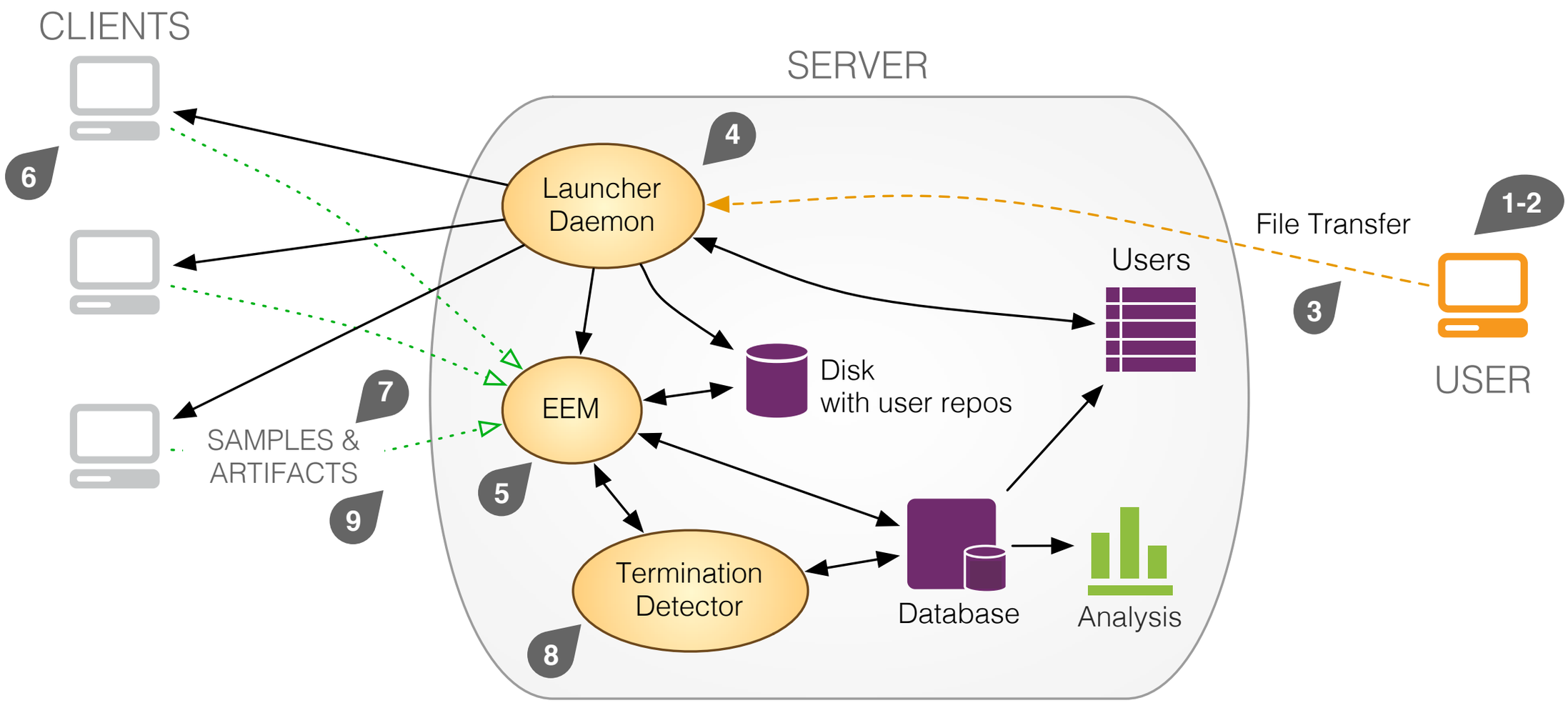
- promote separation of instrumentation code from model code
- match types between producers and consumers of output data more easily
- control dynamically when output data is emitted
- handle structured data from which one field might be of interest (e.g. network packets)
- pre-process data before output
- marshal output data into different formats

Frameworks for ns-3

NSF CISE Community Research Infrastructure
University of Washington (Tom Henderson),
Georgia Tech (George Riley),
Bucknell Univ. (L. Felipe Perrone)



Simulation Automation Framework for Experiments (SAFE)



Perrone, Main & Ward (WSC 2012)

Related work

- Cicconetti, Mingozzi, and Stea (2006)

ns2measure

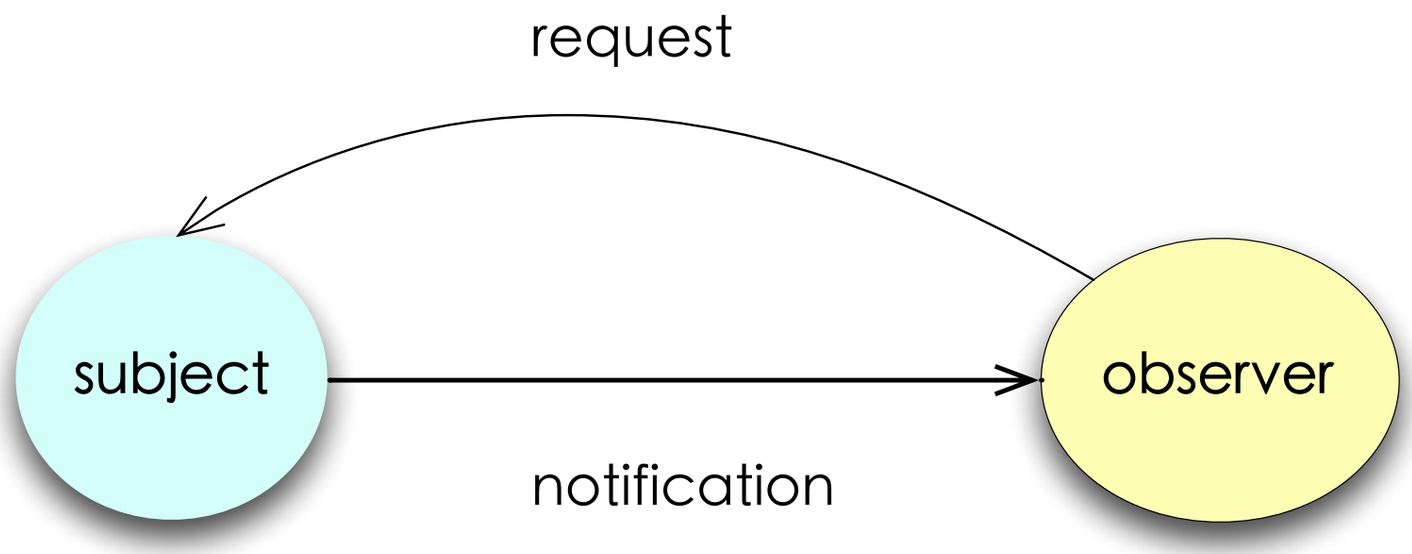
- Ribault et al. (2010)

OSIF

- Helms et al. (2012)

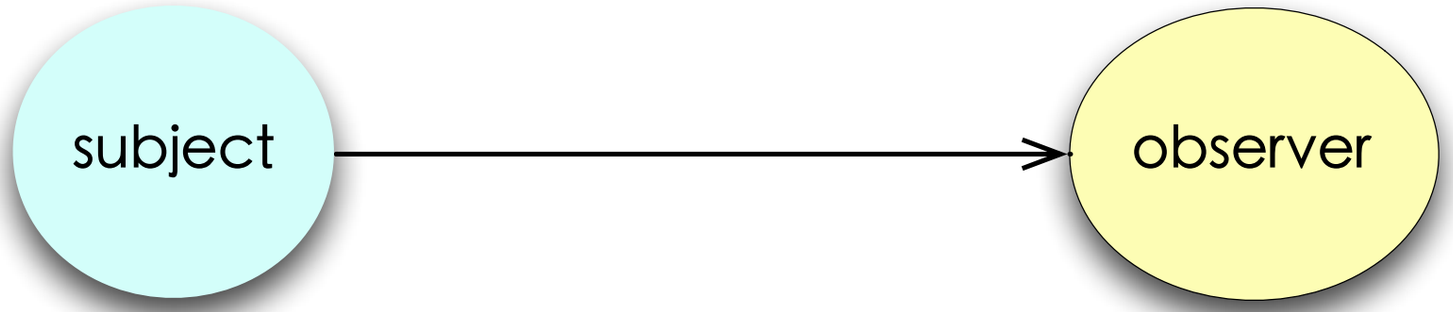
JAMES II

Design



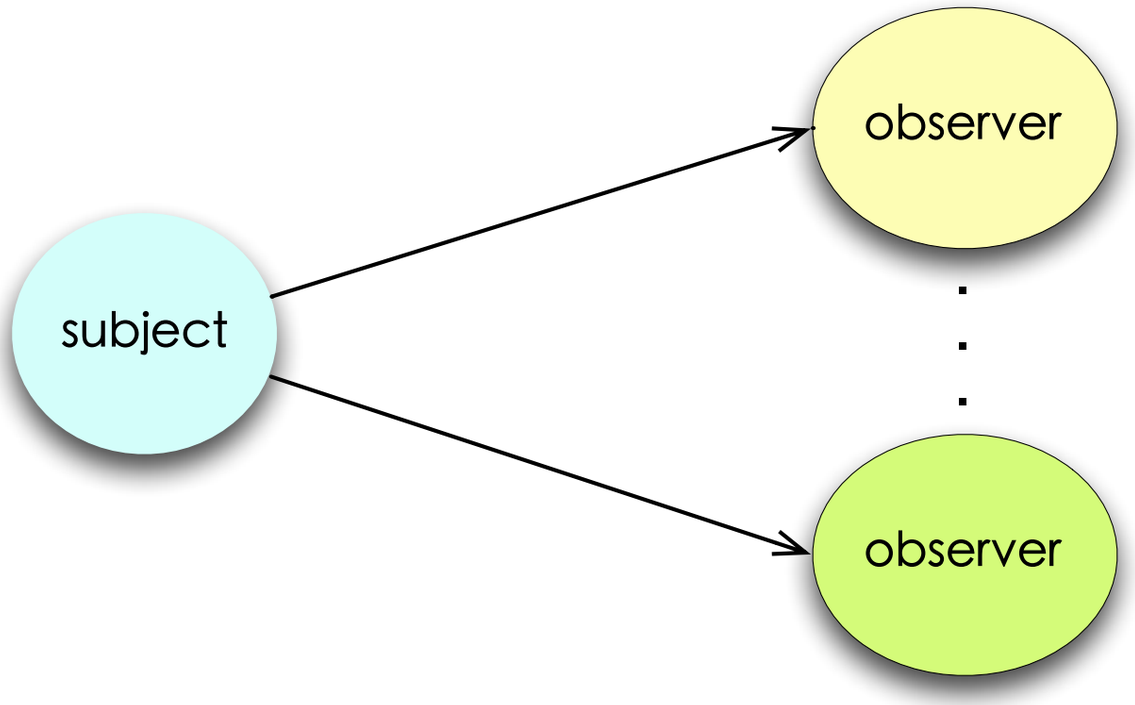
Observer design pattern (Gamma et al. 1995)

Design



Observer design pattern - push model (Gamma et al. 1995)

Design

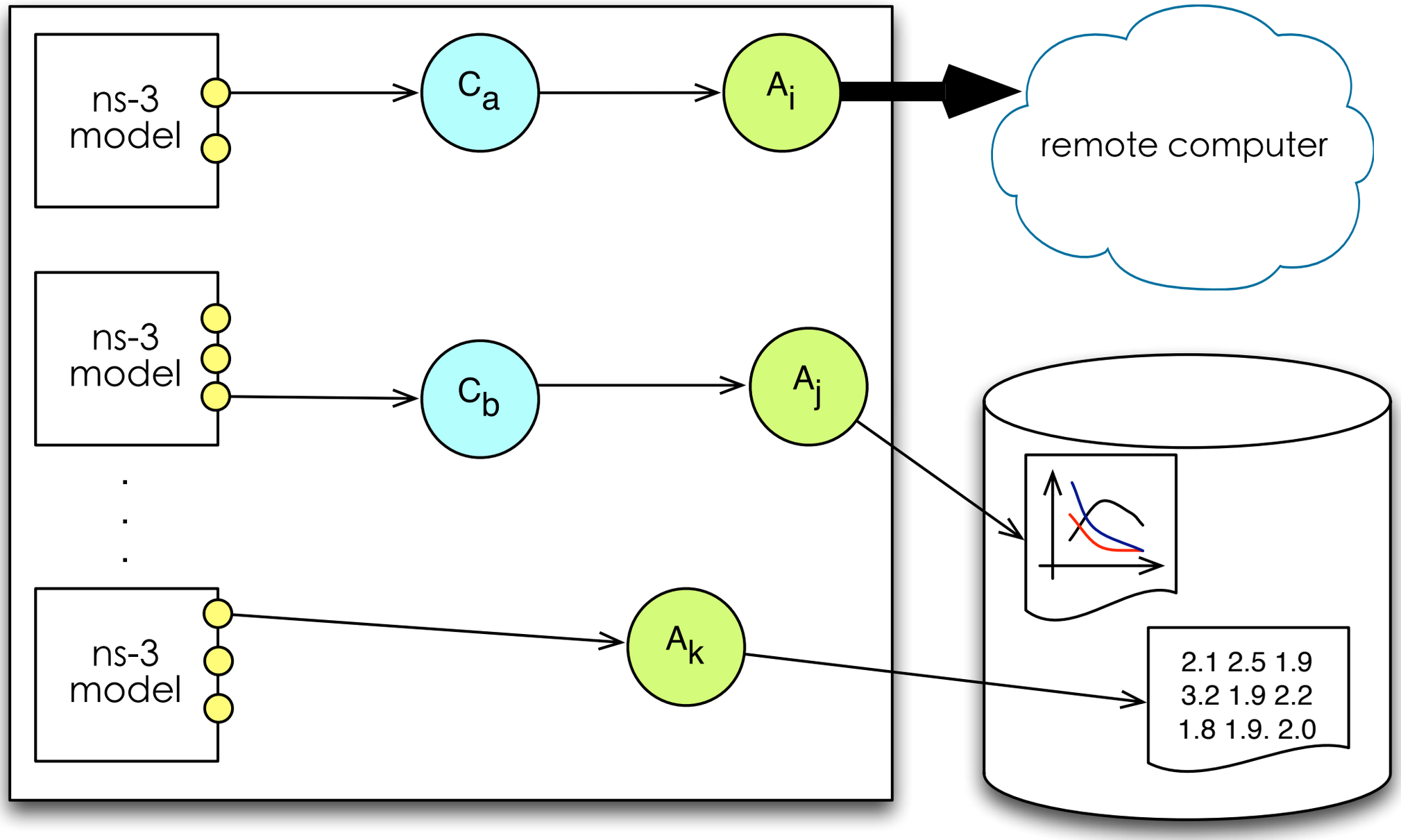


Observer design pattern - push model (Gamma et al. 1995)

Data Collection Framework (DCF)

- DataCollectionObject: base class for DCF elements.
- Probe: extends TraceSources for controllability.
- Collector: encapsulates arbitrary computations on sampled data.
- Aggregator: marshals data into various output formats.

DCF and data flow networks



Low-level API

```
1  // Create the packet probe
2  Ptr<Ipv4PacketProbe> packetProbe = CreateObject<Ipv4PacketProbe>();
3  packetProbe->Enable();
4
5  // Create the collector
6  Ptr<BasicStatsCollector> collector = CreateObject<BasicStatsCollector>();
7  collector->SetPeriodic (Seconds (0.5));
8  collector->Enable();
9
10 // Create the gnuplot aggregator 1
11 Ptr<GnuplotAggregator> gnuplotAgg1 =
12 CreateObject<GnuplotAggregator> ("IPv4_PacketCountPlot");
13 gnuplotAgg1->Set2dDatasetDefaultStyle (Gnuplot2dDataset::LINES);
14 gnuplotAgg1->SetTitle("Packet_Count_vs._Time");
15 gnuplotAgg1->SetLegend("Packet_Count", "Time_(Seconds)");
16 gnuplotAgg1->Add2dDataset("dataset", "Packet_count");
17 gnuplotAgg1->SetTerminal("pdf");
18 gnuplotAgg1->Enable();
19 ...
20 // Hook up trace source with probe
21 packetProbe->ConnectByPath("/NodeList/0/$ns3::Ipv4L3Protocol/Tx");
22
23 // Hook up packet probe with collector
24 packetProbe->TraceConnectWithoutContext("OutputBytes",MakeCallback(&BasicStatsCollector::TraceSinkUinteger32,collector));
25
26 // Hook up collector with gnuplotAgg1
27 collector->TraceConnect ("SampleCount", "dataset", MakeCallback (&GnuplotAggregator::Write2d, gnuplotAgg1));
28 ...
```

High-level API

```
1   // Create the gnuplot helper.
2   GnuplotHelper plotHelper1;
3
4   // Add a probe to the gnuplot helper.
5   plotHelper1.AddProbe ("ns3::Ipv4PacketProbe",
6                         "Node0PacketTxProbe",
7                         "/NodeList/0/$ns3::Ipv4L3Protocol/Tx");
8
9   // Add a collector to the gnuplot helper.
10  plotHelper1.AddCollector ("ns3::BasicStatsCollector",
11                            "Node0PacketTxCollector",
12                            "Node0PacketTxProbe",
13                            "OutputBytes");
14
15  // Configure the plot.
16  plotHelper1.ConfigurePlot ("ipv4-packet-plot-example-packet-count",
17                             "Packet Count vs. Time",
18                             "Time (Seconds)",
19                             "Packet Count",
20                             "pdf");
```

Probes

- Must be connected to TraceSources.
- Allow one to configure the time when data collection starts and stops.
- Provide a method that allows input to be written into the probe object.

Collectors

- Output modes period and asynchronous (batch a number of samples before output)
- Examples include basic statistics, moving averages, batch means, # of mean crossings, MSER-5

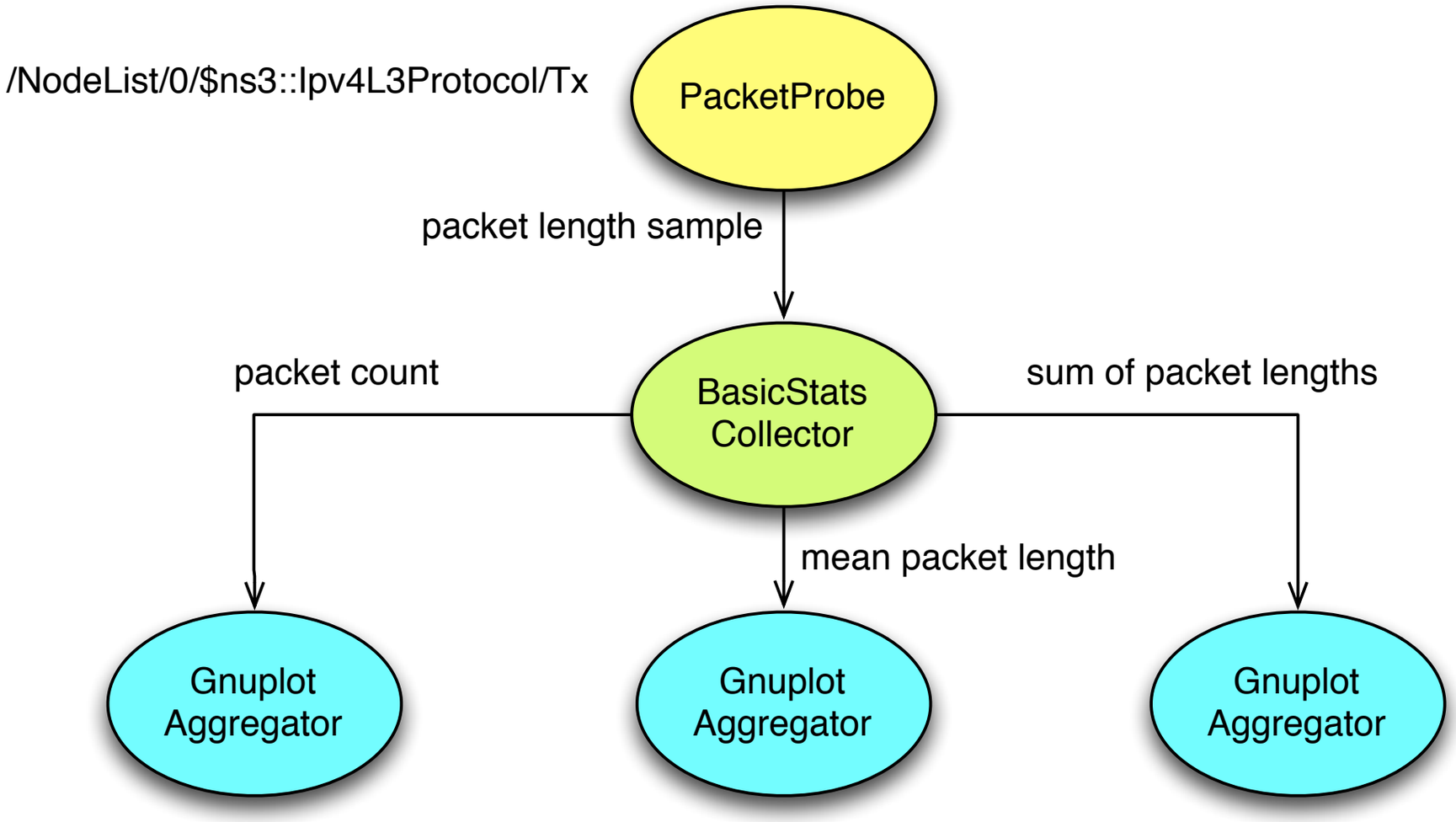
Aggregators

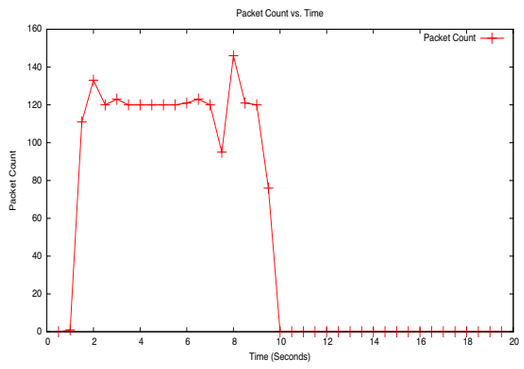
- Receive data and generate output in various formats
- Examples include text file, CSV, SSV, TSV, gnuplot.

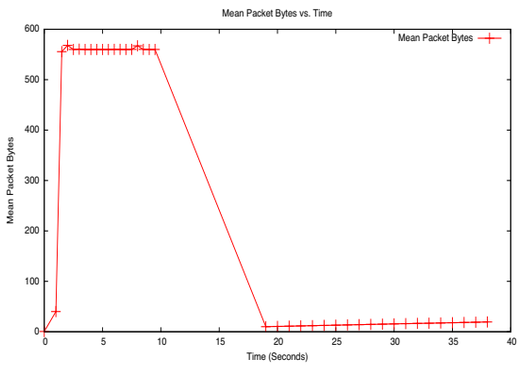
GnuplotAggregator

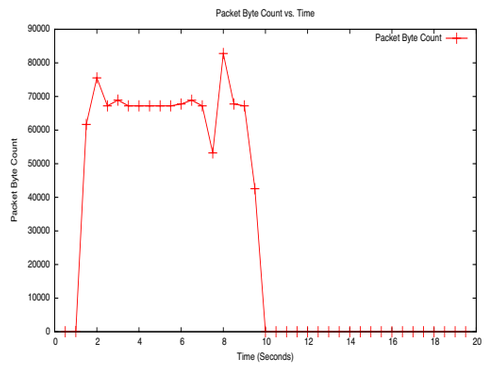
- Addresses basic visualization needs of the ns-3 user (non-interactive plots).
- Built to guarantee basic properties of plots.
- Creates separate files with data and gnuplot script.
- Uses any format supported by gnuplot.

Example with GnuplotAggregator









SafeAggregator

- Interfaces the ns-3 run and a local SAFE process that collects samples to transmit to remote server.

Ongoing and Future Work

- DCF started to appear in ns-3.18 (http://www.nsnam.org)
- Additional functionality in release >= ns-3.20
- Under development: additional collectors (steady-state detection, confidence interval, ...) and aggregators (SQLite, ...)
- Incorporate data provenance functionality

Thanks for your attention!

Questions?