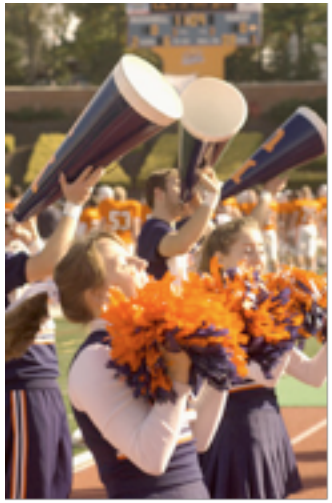
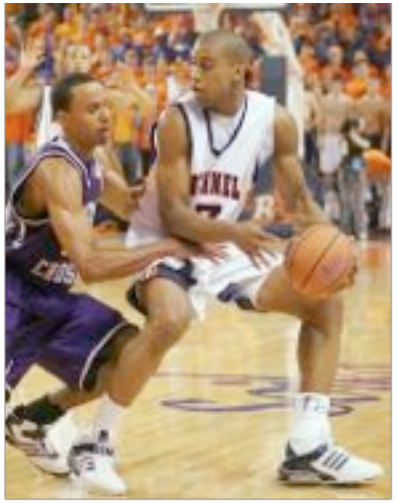
SAFE: Simulation Automation Framework for Experiments

L. Felipe Perrone <perrone@bucknell.edu>
Dept. of Computer Science
Bucknell University, Lewisburg, PA, U.S.A



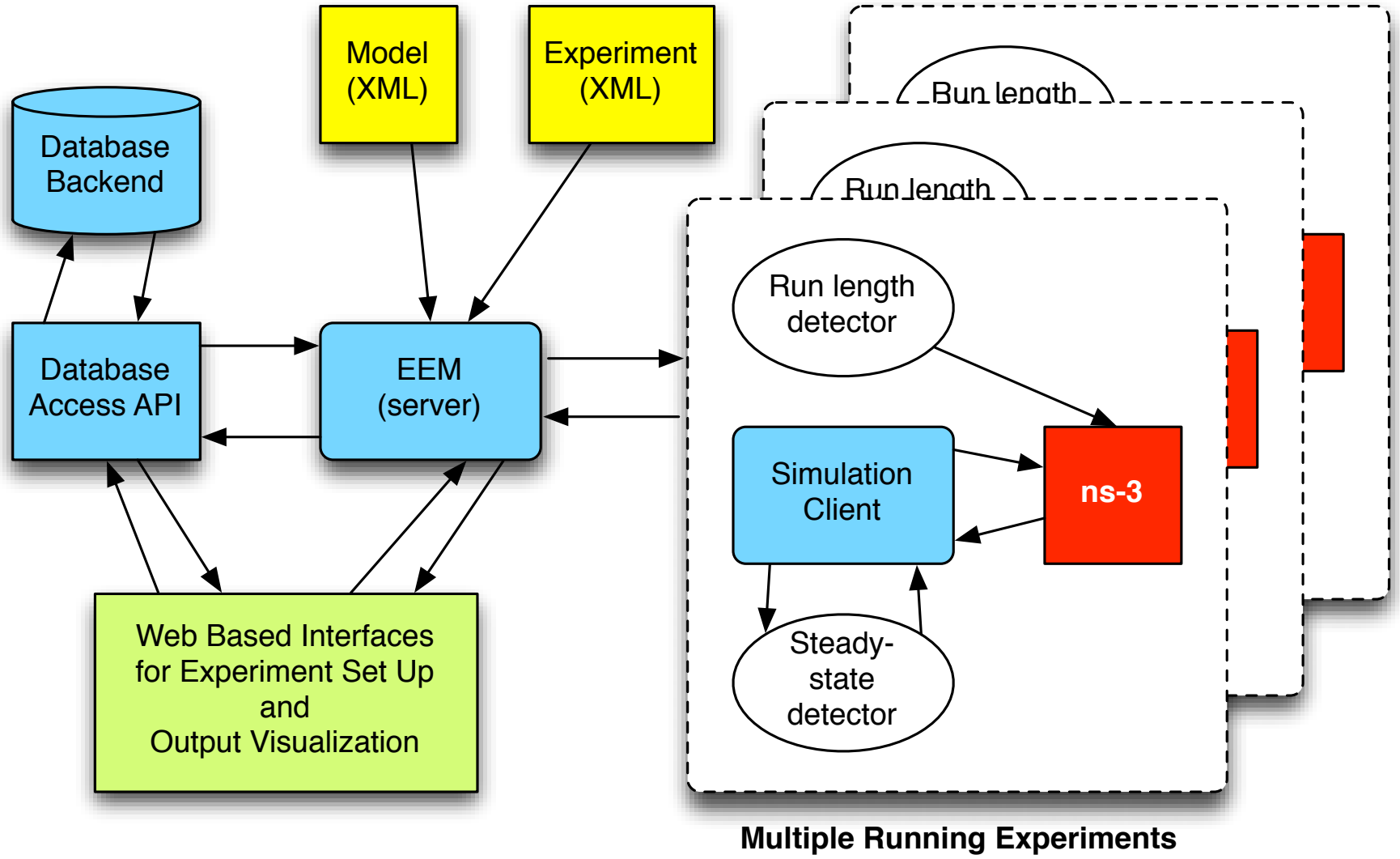
Participants

- Prof. L. Felipe Perrone
- Bryan C. Ward (BCSE ’11)
- Andrew W. Hallagan (BCSE ’11)
- Aurimas Liutikas (BCSE ’12)

Related Work

- SOS: Tim Griffin, Srdjan Petrovic, Anna Poplawski.
- SWAN Tools: Chris Kenna, Bryan Ward, Felipe Perrone.
- ANSWER: Matteo Andreozzi, Giovanni Stea.
- Akaroa 2: Kris Pawlikowski, Don McNickle.

Architecture



Timeline

- First year: major push on XML languages, execution manager and architecture of data collection framework.
- Second year: major push on public release of data collection framework, implementation of web based interfaces, and implementation of steady-state and run length detection.

Experiment Description Language

- **Big picture:** An XML-based language used to define a set of factors and their associated possible levels as lists. These data are used by the framework to construct the experiment design space.
- Provides (so far) two special ways to “prune” uninteresting/irrelevant design points.

Specifying lists of levels

- A "**member-of**" element defines an external list and specifies that a certain factor must take on values only from that list.
- A "**sequence**" element defines a mathematical expression used to create a sequence of level values for a certain factor.

Example: The **member-of** element

```
<member-of>
   <factor>DATARATE</factor>
   <listid>dataRateValues</listid>
</member-of>

<member-of>
   <factor>PACKETSIZE</factor>
   <listid>externalFile.xml</list>
</member-of>
```

Example: The **sequence** element

```
<sequence>
   <factor>DATARATE</factor>
   <test>EQUALS</test>
   <lconst>3000000</lconst>
   <op>MULT</op>
   <rvar>i</rvar>
   <where>
     <range var="i" lo="0" hi="10" delta="1"/>
   </where>
</sequence>
```

Pruning design points

- A "**linking-restriction**" element specifies the factors whose level values must appear in a one-to-one correspondence (e.g., as consecutive pairs).
- An "**exclusion-restriction**" element specifies complete or partial design points which should ***not*** be part of the design space.

Pruning with linking- and exclusion-restriction

```
<linking-restriction>
  <factor>ONTIME</factor>
  <factor>OFFTIME</factor>
</linking-restriction>

<exclusion-restriction>
  <setting factor="ONTIME"
  level="0.0"/>
</exclusion-restriction>
```

Data Flow

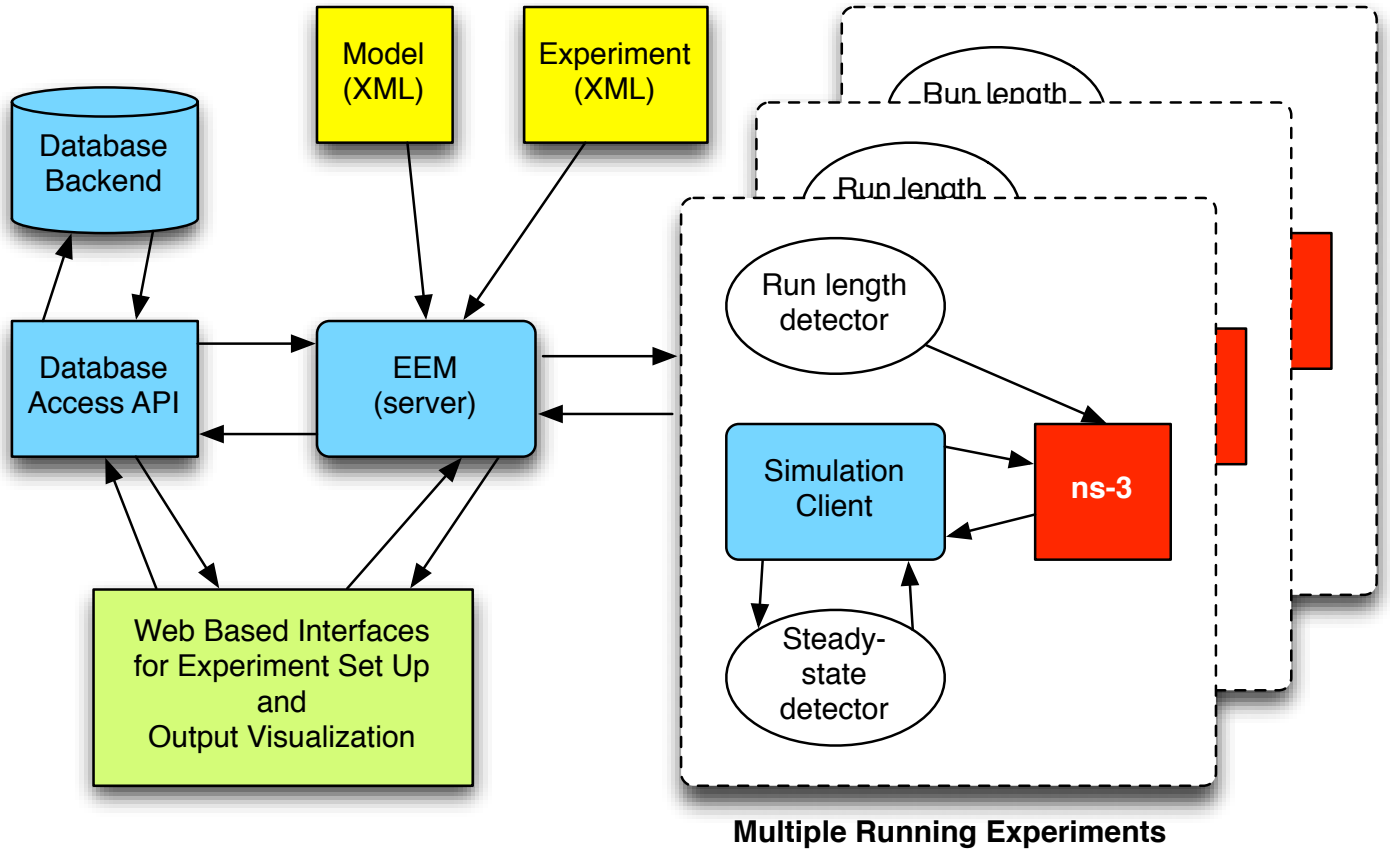
- All *experiment description* documents are validated against a general XML Schema.
- We follow a modular design which specifies APIs between separate tools for validation, parsing, design point generation. This will enable flexibility for further improvements.
- Experiment description documents are parsed by a module which passes to a design point generator the mapping of factors to lists of levels. The design points are used by an *experiment execution manager*.

Experiment Validation

```
awh009@linuxremote1:~/safe/xml$ ./expvalidator.py ../examples/InvalidExperiment.xml -v
    INFO : Found factor 'ONTIME'
    INFO : Found factor 'OFFTIME'
    INFO : Found factor 'DATARATE'
    INFO : Found factor 'PACKETSIZE'
    INFO : Validating <memberof> element where...
    INFO :      <factor> = ONTIME
    INFO :      <listid> = valueList.xml
    INFO :      <listid> matches an internal <levellist> 'id' attribute: True
    INFO :      <listid> matches external filename: False
    INFO : Validating <memberof> element where...
    INFO :      <factor> = OFFTIME
    INFO :      <listid> = internalList
    INFO :      <listid> matches an internal <levellist> 'id' attribute: True
    INFO :      <listid> matches external filename: False
   ERROR : Factor 'NOTAFACTOR' in <memberof> element does not
               appear in the <factorlist>.
    INFO : Validating <memberof> element where...
    INFO :      <factor> = DATARATE
    INFO :      <listid> = anotherlist
    INFO :      <listid> matches an internal <levellist> 'id' attribute: False
    INFO :      <listid> matches external filename: False
CRITICAL : list identifier 'anotherlist' does not match
                   the 'id' attribute of any any internal <levellist>
                   element, nor does it match an external filename in
                   this directory.
    INFO : Validating <memberof> element where...
    INFO :      <factor> = PACKETSIZE
    INFO :      <listid> = internalList
    INFO :      <listid> matches an internal <levellist> 'id' attribute: True
    INFO :      <listid> matches external filename: False
    INFO : Validating <sequence> element where...
    INFO :      <factor> = OFFTIME
    INFO :      <test> = EQUasdfALS
CRITICAL : Sequence test 'EQUasdfALS' is not a valid
                 test.  Must one of ['EQUALS', 'LT', 'GT']
```

Experiment Execution Manager

Client/Server structure written in Python, based on the Twisted network programming framework.



Experiment Execution Manager

- The server processes experiment descriptions and generate design points, which are dispatched to clients.
- Clients run ns-3 simulations for design points, which use a **data collection framework** to generate samples of metrics. Results are sent to external processes for steady-state and run length detection, then back to server for storage in SQL database.
- API for accessing the results is in the works.

Simulation Client

- Requests a design point to run from the EEM.
- Executes the simulator with that design point.
- Listens for samples from the simulator via a pipe.
- Reports samples to the EEM.
- Listens for further instructions from the EEM to decided when to terminate the simulator (via a signal).

Data Collection Framework

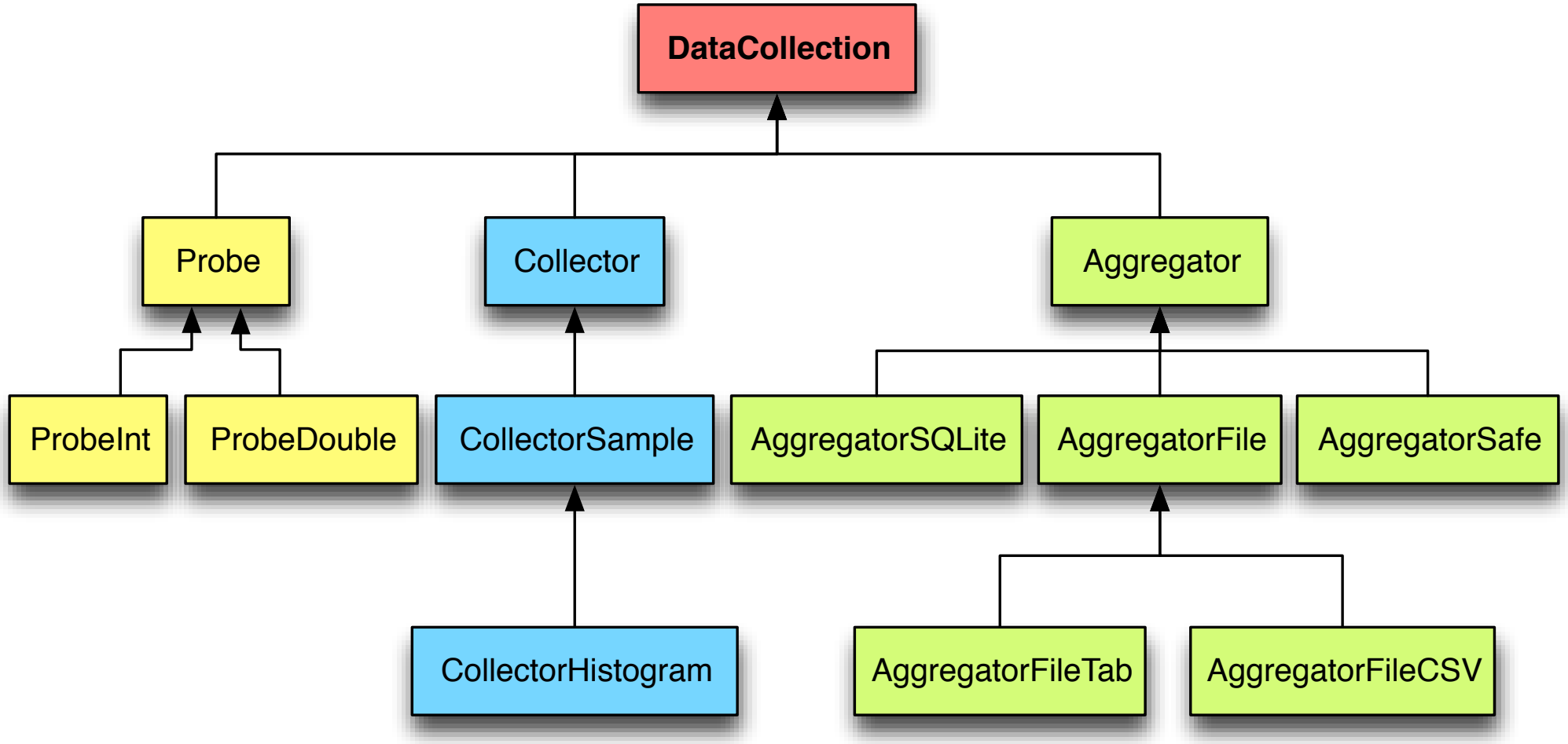
Goals:

- Record ‘samples’ of variables (attributes and non-attributes) every time there’s a change in their value.
- Tag data with timestamp and an identifier of the source (context).
- Compute basic statistics on samples.

New Classes Defined

- **DataCollection:** Base class for all elements of the data collection framework.
- **Probe:** Mechanism for detecting changes to a variable.
- **Collector:** Contains and processes samples generated by a probe.
- **Aggregator:** Sends samples to chosen output according to a pre-defined format.

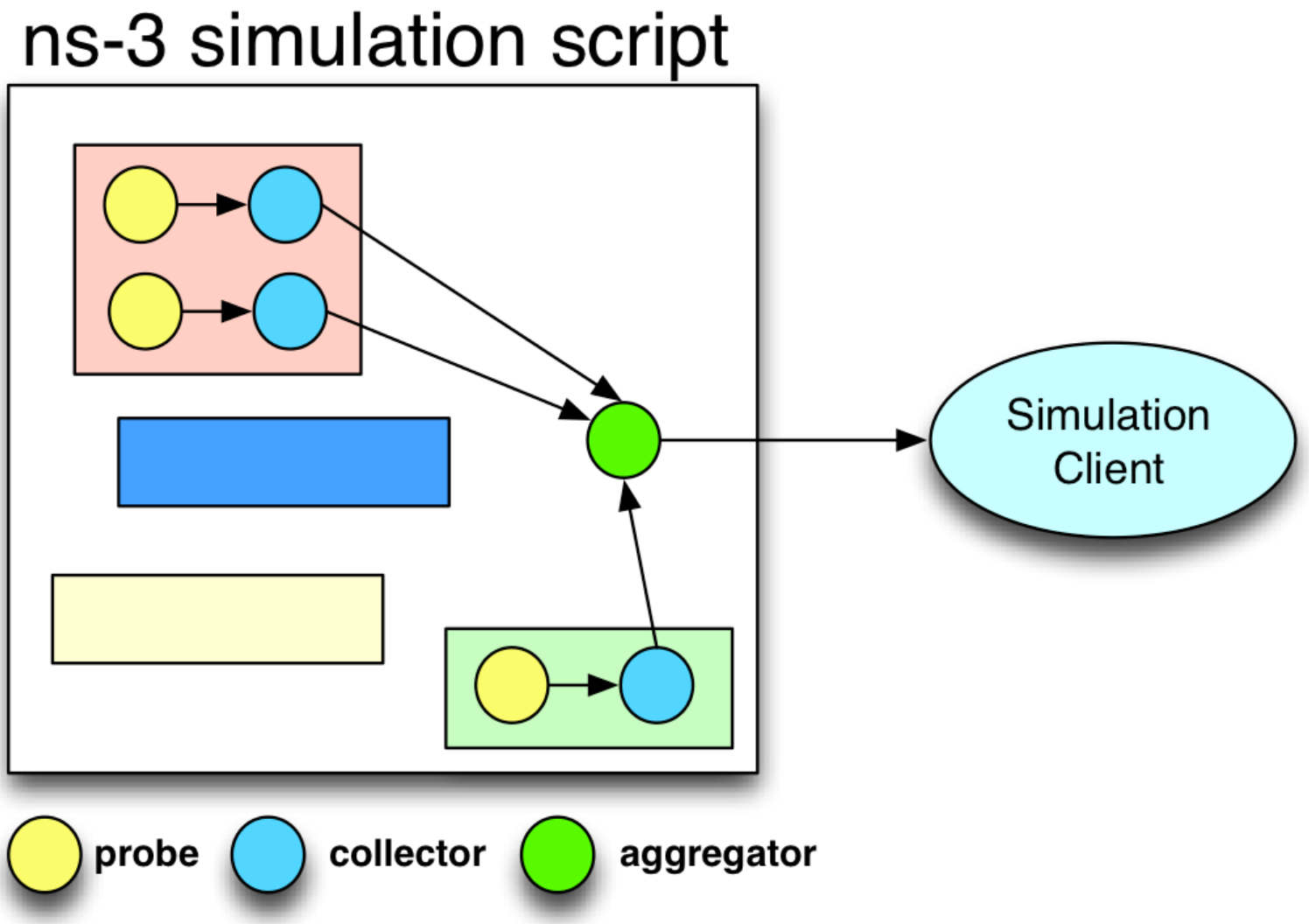
Data Collection Framework Classes



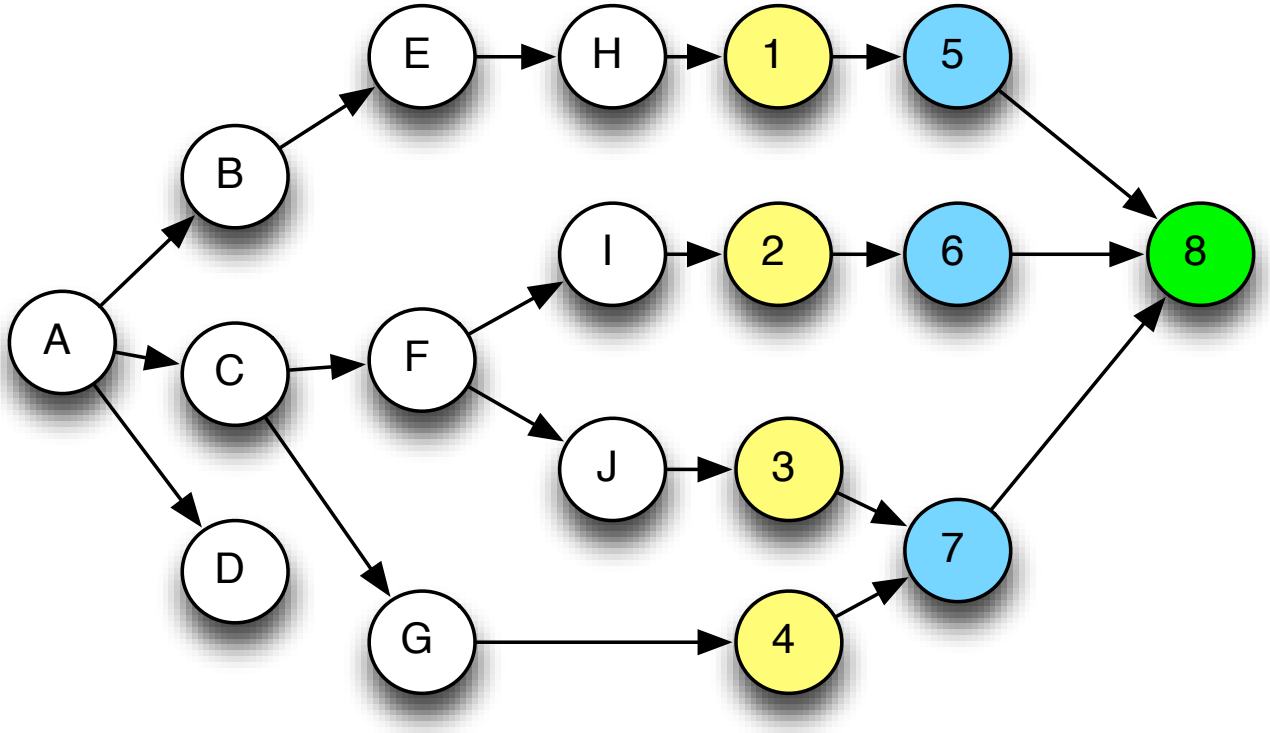
DataCollection

```
class DataCollection : public Object {

public:

   static TypeId GetTypeId ();

   DataCollection ();

   virtual ~DataCollection ();

   virtual bool GetStatus () const;

   virtual void SetStatus (bool s);

private:

    bool m_enabled;

    typedef std::map<std::string, Ptr<DataCollection>> DataCollectionMap;

    DataCollectionMap m_inputs;

    DataCollectionMap m_outputs;
```

Usage Example



Context / Identification



/A/B/E/H/1/5/8

.
.
.

Points to Notice

- ns-3 defines the TracedValue template class - when a variable changes, a pre-determined function is called. Some ns-3 classes use TracedValue to define trace sources, which can be connected to trace sinks via Config::Connect (one identifies the source using a path to the right object and the callback to serve as sink).
- The value monitored by a probe is not an attribute; it can be “just a variable” in the scope of some method (main use case).

Control Requirements

Global disable: No samples; negligible run time cost.

Global enable: All probes report samples during a window of simulation time specified by a start and an end value. Outside this window, no samples are reported.

Local enable: Only individually selected probes report samples during a window of simulation time.

Data Type Requirements

- **Integer:** A standard integer data type (64 bit?)
- **Double:** A standard double data type.

- **Scalar:** The probe generates scalar data types.
- **Non-scalar:** The probe generates a data type that can be seen as a collection of scalar values (e.g. a vector of values).

Milestones

- April: data collection framework, execution manager, and language tools out for review.
- May/June: development of interface components, analysis and graphing tools, modules for steady-state and run length detection.

Project Web Resources

- Current project site:
  http://redmine.eg.bucknell.edu/perrone/projects/framework
- Upcoming revamped project site (summer 2011):
  http://redmine.eg.bucknell.edu/safe
- Data Collection Framework code review (under refactoring):
  http://github.com/lfperrone

Acknowledgements

- Pavel Boyko, IITP RAS
- Mathieu Lacage, INRIA