Simulating Routing Schemes on Large-Scale Topologies

Luc Hogie, Aurélien Lancin, Issam Tahiri
Dimitri Papadimitriou

DCR closing meeting / Paris, France

6/12/2010

Outline

Tools - Java4Unix

Java4Unix

`http://www-sop.inria.fr/members/Luc.Hogie/java4unix/`

- generic install script from Java4Unix project.
- automatically download and install drmsim into $HOME/.drmsim,
- PATH and CLASSPATH automatically set to use DRMSim commands,
- start drmsim like UNIX software.

Tools - EPR

EPR - Eclipse Project Releaser

`http://www-sop.inria.fr/members/Luc.Hogie/epr/`

- release project developped from Eclipse,
- copy JAR archive to the specified repository (used with j4u for installation)
- tag release into subversion repository,
- manage dependencies.

How to use DRMSim

Hierarchical configuration file:

- defines section, e.g. simulation, model, metrics, scenario, etc.
- allows modularity,
- easy to parse and export to other format.

```
section graph
    number-of-nodes={10000, 20000, 30000, 40000}
    section prng
        seed={0.6}
    end of section
end of section
```

How to use DRMSim?

DRMSim is composed of

- Dipergrafs,
- Mascsim,
- DRMSim as a model.

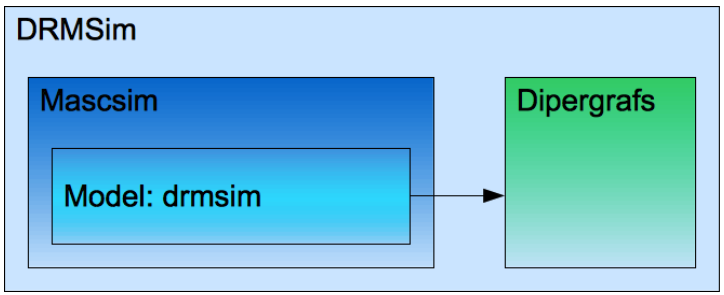


Figure: The software modules used in the development of DRMSim.

Main command to execute DRMSim:

```
# drmsim [options] config-file
```

Execution scheme

Monolitic:

- sequential, all simulation instances are executed one after one. Results are stored until all simulations finish.
- parallel, one simulation instance is executed per core. If they are more instances than cores, they are queued.

Distributed:

- job distribution, simulation instances are distributed among available peers (daemons) in the network.
- distributed simulation, simulation model is distributed among available logical processes. LPs are managed by daemons.

Registering metrics

Simulation is instanciate from the class name provides in the configuration file. A simulation is composed of a system (e.g. topology).
When simulation is initialazing, first, metrics are loaded from configuration file.

- metrics are registered into the system and the simulation.
- for each event, the system or the simulation informs listening metrics of the event.
- metrics computation must be optimized. Computation time directly impact simulation performance.

System instanciation

System is initialized by the simulation model. It provides system entity factories to external library.
For DRMSim, router and link factory is provided to dipergrafs to customized generated graph.

Dipergrafs allows to:

- generate topologies,
- save/load topologies to/from files.

When creating a graph, Dipergrafs compute its properties and store them in a cache. It allows to reduce metric computation time during simulation.

Dipergrafs cache will be an important feature for dynamic model.

System instanciation

Once topology created, routing tables can be loaded from a file defined in the configuration file.

Routing tables can be loaded for:

- BGP,
- NSR.

Routing table can be saved into a file once the simulation is terminated.

Initial event

In order to allow message routing, protocol initial events must be scheduled.
For each router, method `initalEvent` is called. Protocol table initialization begins.

Following the scenario, simulation will stop:

- for routing scenario, if
  `the number of message routed + the number of routing failure = the number of scheduled routing message`,
- for BGP convergence, if no more MRAI event is scheduled,
- for BGP simpler, if no more UPDATE event is scheduled.

The termination condition can be override depending of the needs.

Message routing

Two kinds of event are used:

- Message event, next hop is computed by method `computeForwardLink`. Must be optimized, heart of the simulator!!!
- Protocol event:
  - Message, internal protocol event.
  - Timeout. internal timeout event.

  Internal protocol event knows its final router destination. No routing and next hop computation are performed, only direct method calls.

Splitting event in different classes allows fined grained measures.

Measures

Metrics listen the simulation and the system. Simulation and system forwards events to concerned metrics.

To avoid to store to much measures, interval storage can be defined. Metric always computes the measure, but store it depending of the interval.

An initial and final measure is always computed.

Two kinds of metrics are used:

- Traffic metrics (stretch, number of routing message, number of routing failure, etc).
- System metrics (size of routing tables, number of router failure, etc).

Internal metrics

Traffic metrics:

- number of routing messages,
- number of induced routing messages,
- number of protocol messages,
- total number of messages,
- number of routing failures,
- number of timeouts;
- stretch.

System metrics:

- number of router failures,
- number of link failures,
- size of routing tables,
- number of entries of routing tables,
- number of exhaustive routing tables.

Protocol specific metrics

It is possible to plug own protocol metrics. First, protocol events must inheritate from DrmEvent and implement one of the following interfaces:

- InternalProtocolEvent,
- InternalMessageProtocolEvent,
- InternalTimeoutEvent.

The source router must be provided to DrmEvent. The event must call the system `fireProtocolEvent` method in order to wake up listening metrics. The metric id is given when calling `fireProtocolEvent`, so concerned metrics can identified the event.

Measure restriction

It is possible to focus measures on a subset of nodes or links. Selection of concerned entities are done by a class provided in the configuration file. This class switch monitoring router attribute to true.

Configuration example:

```
nodes-to-monitor =org.drmsim.metric.NodeSelectionByDegree
nodeByDegree= >160
```

Processing results

When a simulation terminate, a result is provided.
First, run's results are merged. The metrics averages and standard deviations are computed.

A specific class must be provided in the configuration file to compute specific measures. (e.g. statistical measures of a metric by number of nodes).

Measures are plotted using Gnuplot. Processes are excuted in parallel given the number of cores availables.

Independent routing scheme modules

A routing scheme can be composed of some basic algorithms:

- DRMSim provides basic algorithms, called modules, easy to reuse inside a routing scheme;
- modules are tested and validated.

Modules control:

- possibility to provide a self data structure;
- possibility to catch and control module's events.

Independent routing scheme modules

Modules provided:

- distributed flooding algorithm;
- distributed Bellman-Ford algorithm;
- distributed Dijkstra algorithm;
- landmarks based routing algorithm;
- vicinity balls based routing algorithm;
- naming based routing algorithm;
- interval based routing algorithm.

Basic routing data structure

Modules provided:

- index routing table;
- array routing table;
- double indexing routing table;
- hash routing table;
- neighborhood table;
- label entry;
- interval entry;
- etc.

Problems

- A routing event scheduled before the convergence may lead to a routing failure.
- Be able to decide when to save routing tables.
- Metric model use a lot of memory with little intervals.
- Basic modules, some bugs remains concerning modularity.
- AllToAll scenario is time consuming.

Future tasks

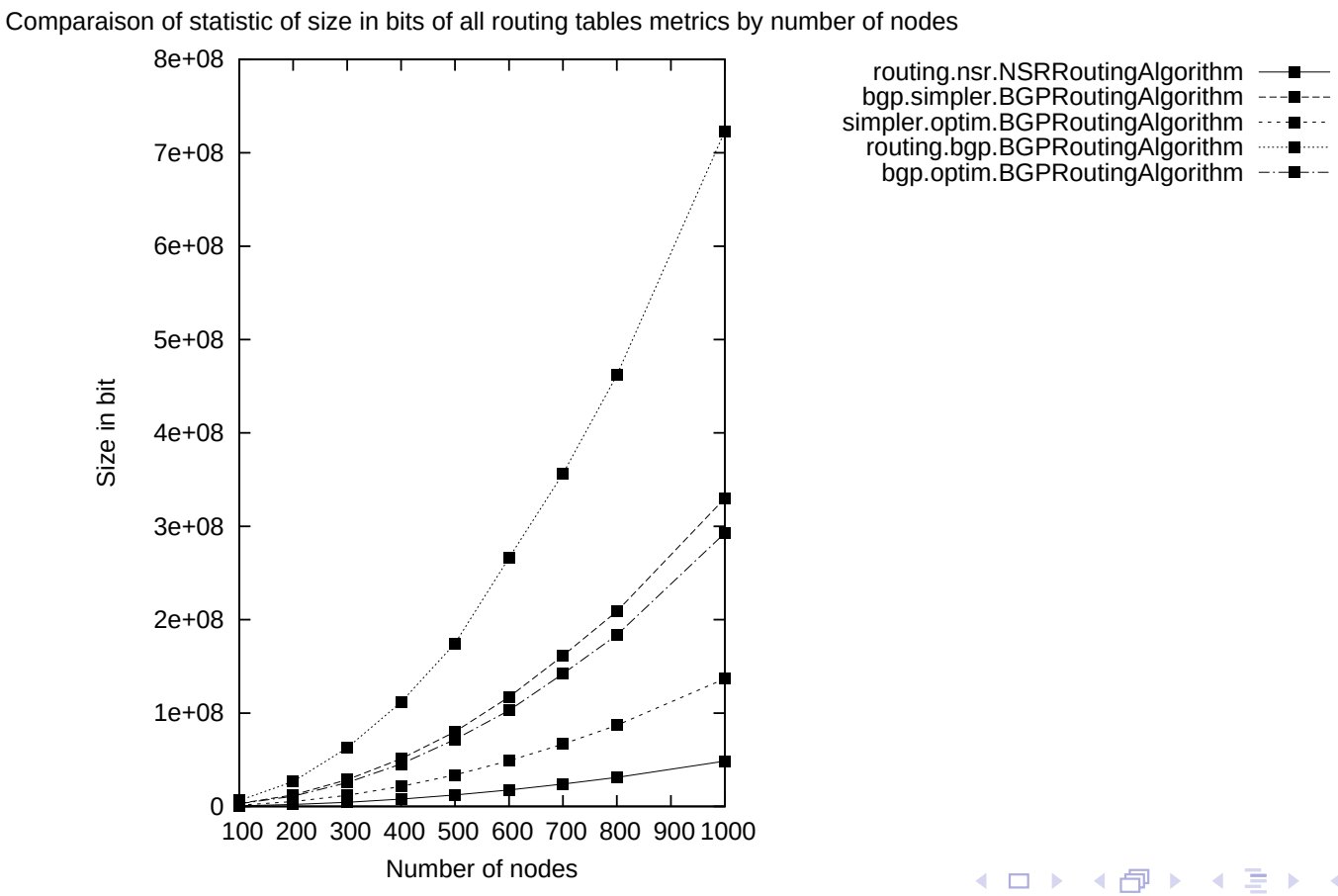
- Improving DIPERGRAFS properties cache. Useful for dynamic model.
- Measure interpolation to reduce memory usage.
- Be able to provide a set of topology to load with their corresponding protocol.
- Log threading.
- Threading measure saving to file.

Next steps:

- policy model,
- dynamic model.

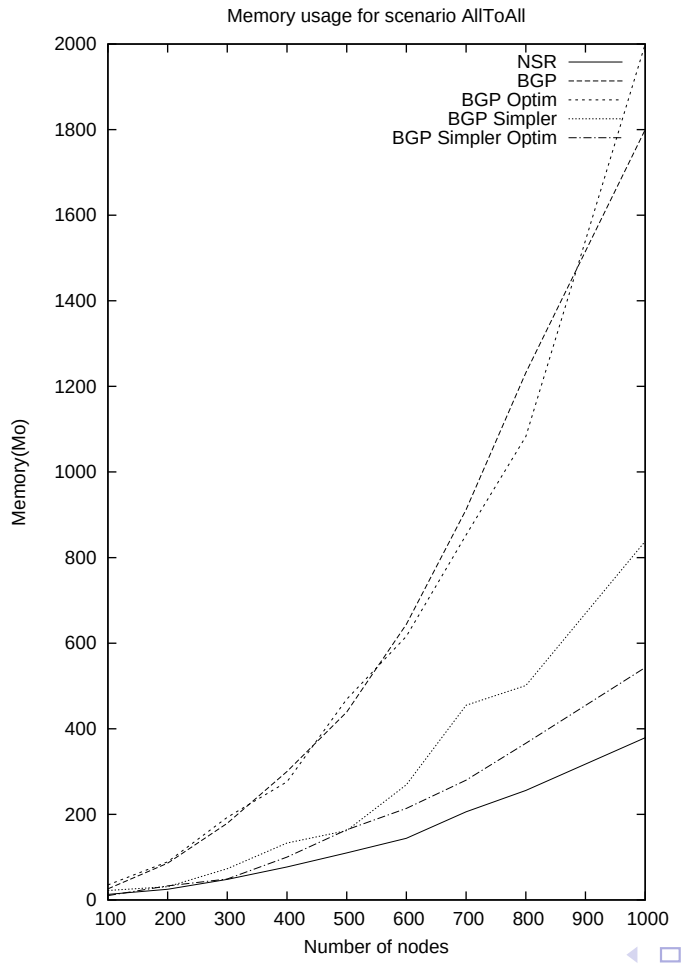
NSR and BGP routing tables size

Simulation on GLP topology from 100 to 1000 nodes



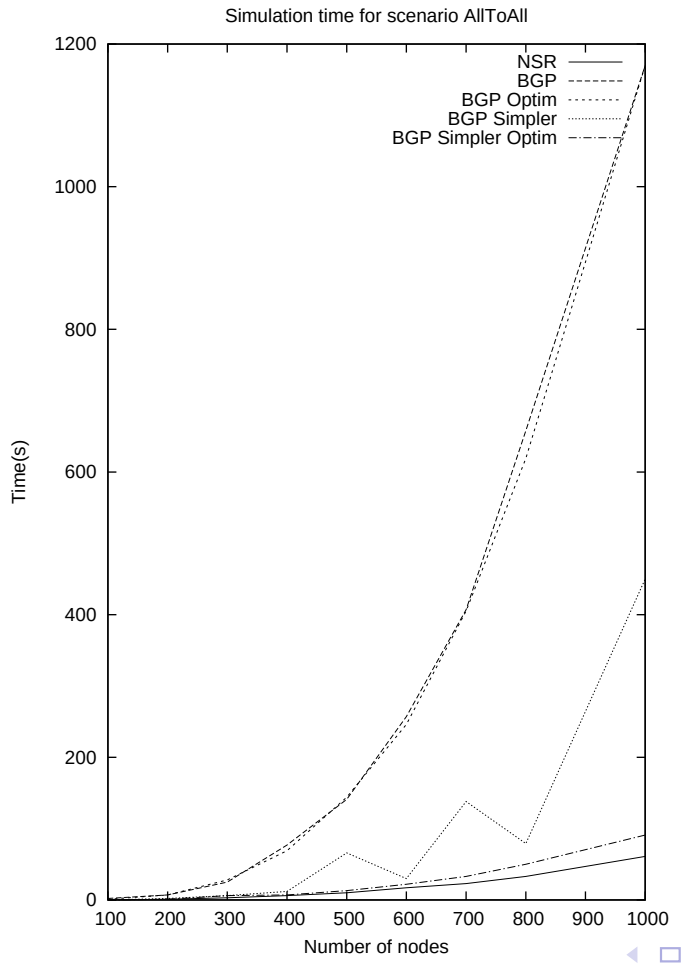
NSR and BGP memory usage

Simulation on GLP topology from 100 to 1000 nodes



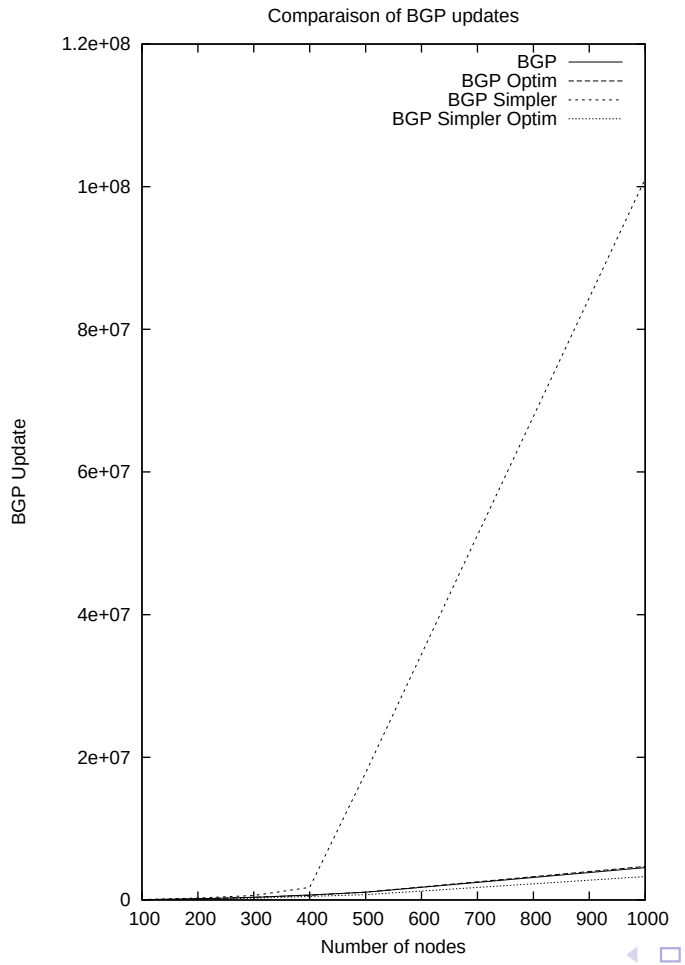
NSR and BGP simulation time

Simulation on GLP topology from 100 to 1000 nodes



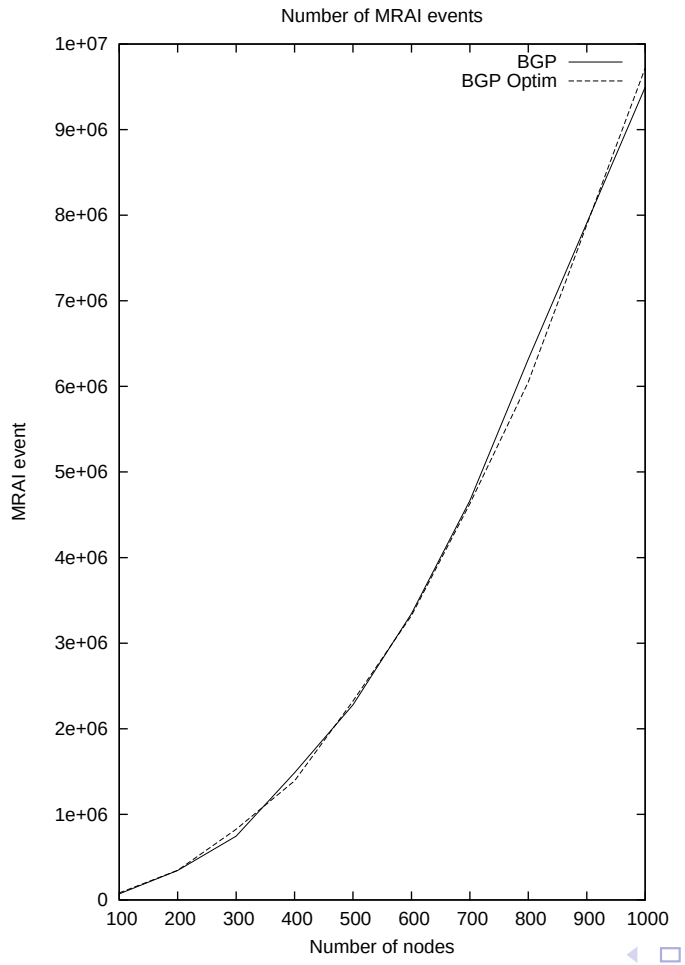
BGP updates

Simulation on GLP topology from 100 to 1000 nodes



BGP MRAI

Simulation on GLP topology from 100 to 1000 nodes



BGP simpler optimized on 20 000 nodes

```
Scenario OneToOne with 2000 routing.
Simulation time : 3 hour(s), 1 min(s), 57 second(s)
Memory used: 43 GiB

Number of routing messages delivered: 2000
Number of expected routing messages to deliver: 2000
Number of induced routing messages: 6922
Total number of routing messages: 8922
Number of internal protocol messages: 1850626537
Total number of routing tables entries: 400000000
Additive stretch: 0.0
Multiplicative stretch: 1.0
Size of routing tables: 1503272960
```

Distributed DRMSim

Only conservative methods seem ok.

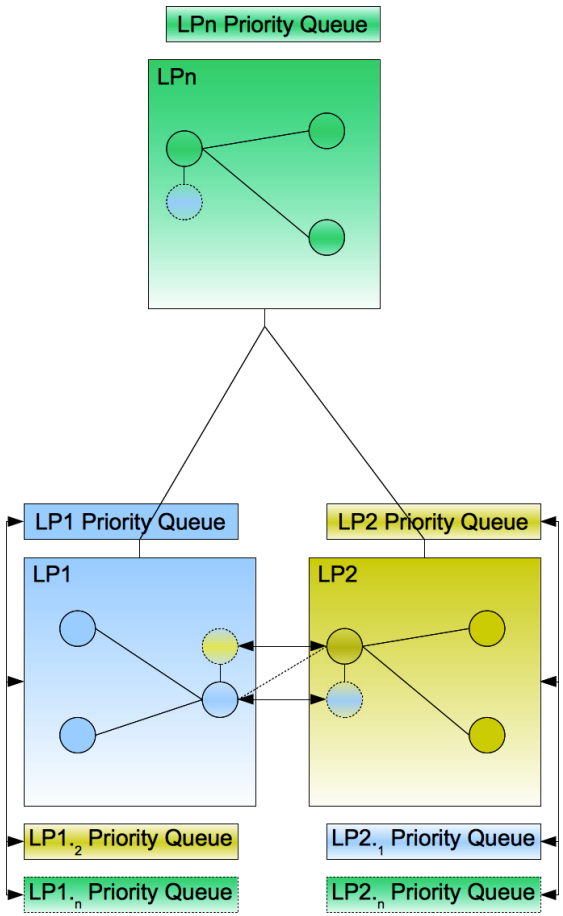
- first problem, graph partitionning,
- second problem, system modelisation,
- third problem, lookahead.

For a first draft, the proposed architecture is composed of:

- a simple manual graph partitionning,
- two daemons for simulation management,
- rmi based network communication.

Distributed system

- Each LP has a copy of all its influencing LPs.
- A Border router compute locally the information and then sends them to its copies.
- A event scheduled to another LP will be executed on both LPs.



Thank you

Questions?