

A Simulator for the Investigation of Dynamic Compact Routing over Large Networks

Luc Hogue, Issam Tahiri

Outline

- 1 Currently state of operation of Internet
- 2 Description of the Research project
- 3 Motivations for building a new custom simulator
- 4 MASCSIM: a distributed discrete-event simulator
- 5 ALUSIM: a routing simulator based on MASCSIM
- 6 A few words about the implementation of the whole thing
- 7 Work in progress

Internet currently comprises about 30k AS and about 300k IP routes. Growth rate: 1.2-1.3/year. The number of routing table entries:

- was 175k in 2006;
- is going to be 300k by the end of 2009;
- is likely to reach 0(1M) in 5 years.

BGP (Border Gateway Protocol) [Y. Rekhter and (Eds), 2006] **already exhibits** stability and convergence issues. It proves unable to manage tomorrow's Internet. Internet was born from Research. More Research is now needed to achieve larger scale.

Joint project, funded by Alcatel-Lucent-Bell, involving Mascotte and the LaBRi whose the aim is to provide routing protocols for the backbone Internet which are in terms of:

- the **size of the routing table** they build;
- the **stretch** they results in;
- **traffic messages** (routing updates) they generate;
- their processing time **complexity**.

Proposed routing protocols will be investigated though simulation: we want a simulator enabling the investigation of **compact-routing protocols for large dynamic networks** (i.e. in the order of ten thousands nodes).



MASCOTTE



Our motivation for writing a custom simulator lies in our needs for:

- implementing tweaked versions of standard routing protocol;
- defining and implementing specific topology generators;
- adjust the granularity in order to achieve large-scale simulation.

No currently available simulator would able us to meet this needs simpler than by developing our own solution.

ALUSIM is not built from scratch. It is derived from the MADHOC simulator.

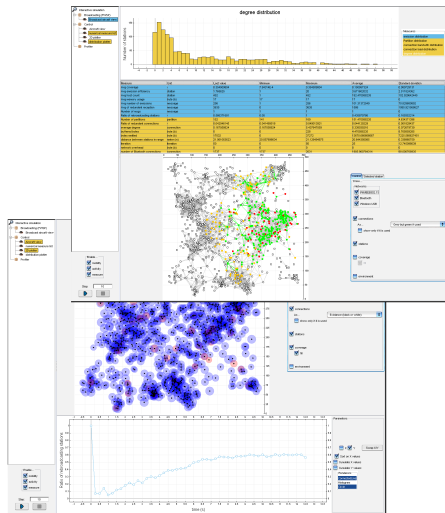
MADHOC is a simulator for mobile wireless networks. It was developed in 2003-2007 and was used by 10 labs for many different purposes (broadcasting, protocol optimization, graph generation, etc).

We opt for reusing MADHOC because:

- it's a lot of work that can be reused;
- we have a deep understanding of its internals;
- it proved efficient and stable;
- it is easy to extend/modify.

MADHOC's [Hogie et al., 2006] aims to provide:

- models for **heterogeneous nets** (Wi-Fi + Bluetooth, etc);
- **metropolitan mobility** models;
- the ability to simulate **large numbers of nodes**;
- a fast simulation engine;
- both **GUI and console** mode.



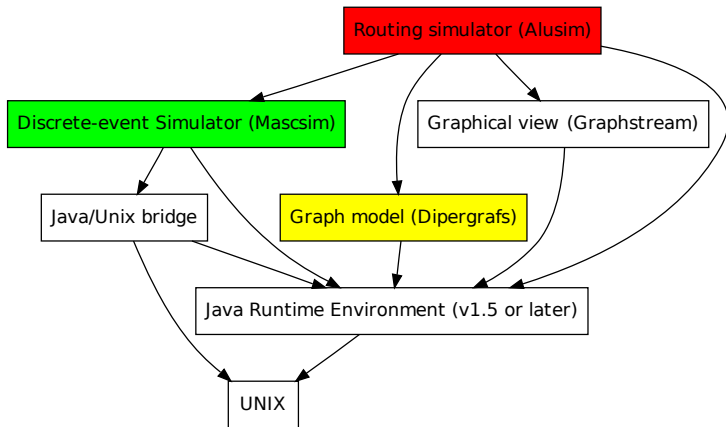


Figure: the software modules used in the development of ALUSIM. Arrows represent module dependencies.

- MASCSIM is an object-oriented discrete-event simulator. In MASCSIM, an event is an object whose the insertion into the event queue means: "it's going to happen **this** at **that time**".
 - this** is what the event consists in, its code;
 - that time** is the occurrence date of the event.
- The simulation starts with a number of user-defined initial events. When they execute, events schedule other events.
- MASCSIM does not use a *termination-event* to complete the simulation. Instead it relies on a user-defined termination condition (boolean function).

The input file for the MASCSIM simulator is **declarative** (no instructions there!). It is a set of

$$key_i = \{value_0, value_1, \dots, value_n\}$$

entries which define the parameters for the simulation. This is an example:

```
number_of_routers={50}
glp_beta={0.9}
inter_step_duration_millis={2000}
simulation_class={inria.alu.AluSimulation}
number_of_links={100000}
metrics={inria.alu.metrics.NumberOfRoutingFailures,
         inria.alu.metrics.NumberOfDeliveries, inria.alu.metrics.Stretch}
topology_generator={inria.dipergrafs.topology.GLPTopologyGenerator,
                   inria.dipergrafs.topology.SymmetricTopologyGenerator}
system_listeners={none}
root-of-tree={0}
routing_protocol={inria.alu.routing.ShortestPathRouting}
prng_seed={0}
glp_stepProbability={0.7}
number-of-routings={10}
valid-values-minimum-number={3}
```

An instance of a simulation defines a set of **metrics** (for instance number of nodes, stretch, etc). To every metric is associated a set of **timestamped measures**. These sets are fed all along the simulation process. They constitute the result of the simulation.

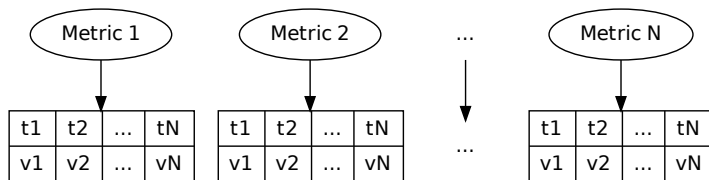


Figure: The binary result file aggregates the set of measures associated to every user-defined metric.

MASCSIM design is based on the following principle: simulation results are valid if and only if they were statistically computed out of the results of numerous simulations. In other words, The set of simulation processes can be seen as a pool of **independent jobs** to be executed.

For example, comparing 3 protocols on n nodes with $n \in [10^2, 10^4]$ gives $3 \times 100 \times 30 = 9.000$ jobs.

MASCSIM defined 4 ways of computing these jobs:

- **sequentially**;
- in parallel **threads**;
- in parallel using RMI-enabled **workstations on the LAN**;
- in parallel using a dedicated **computational grid**.

- If configured to use Java Remote method Invocation (RMI), MASCSIM **scans** the local area network (LAN) in search of computers running MASCSIM. This RMI-based solution is **decentralized**.
- Because of the nature of RMI, which was designed to operate on trusted network environment, MASCSIM's RMI operation mode can not deal with firewalls and NAT.

If configured to use its dedicated grid, MASCSIM connects to a server where it:

- post simulation jobs (in the form of input files);
- periodically checks the availability of corresponding result files.

By their side, grid peers will in turn connect to the server in order to:

- download simulation jobs requests;
- post result files, once they computed them.

MASCSIM deployment uses a method inspired from **network booting**.

Be it on RMI-based clustering or Grid, **MASCSIM is not installed on the network peers**. Instead peers simply run a **bootstrap** application which contains the bare minimum to identify, download and execute the last version of the simulator.

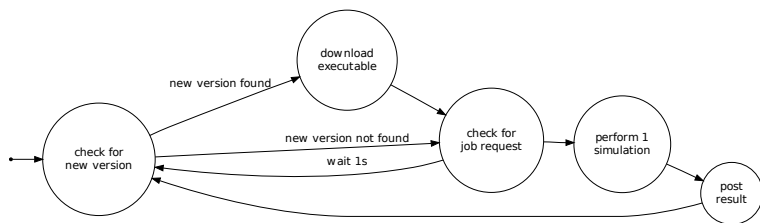


Figure: This mechanism ensures that **new releases of the simulator will be automatically deployed** on all peers.

In order to avoid the unnecessary recomputation of same input files, MASCSIM uses **result caching**.

- It computes the hash-code of input configuration (not the input file!).
- This hash-code is used as the prefix name for the input file.
- The output file, which uses the same prefix, is stored on a global web-accessible repository.
- It is then easy to obtain available results without any computation.

MASCSIM can be used through a set of **command-line applications**.

- `mascsim-campaign`
- `mascsim-find-out-input-file-name`
- `mascsim-result-info`
- `mascsim-result-plotter`
- `mascsim-simulator`
- `mascsim-develop-config`
- `mascsim-lan-scanner`
- `mascsim-result-merger`
- `mascsim-rmi-lan-scanner`

The network model relies on the concept of directed hypergraphs, which is the most adequate data structure when it comes to represent networks. DIPERGRAFS implementation relies on four coupled incidence lists. I/O are in $O(1)$.

- $n \rightarrow$ *set of out edges*;
- $n \rightarrow$ *set of in edges*;
- $e \rightarrow$ *set of tail nodes*;
- $e \rightarrow$ *set of head nodes*.

Node has a integer ID from 0 to n . Upper limits for the numbers of nodes and edges have to be defined.

The following topology generators have been developed:

- Set of basic schemes (tree, grid, chain, random, etc)
- front-end of Inet [Jin et al., 2000] (Inet must be installed);
- bridge to Brite [Medina et al., 2001];
- extraction of topologies from CAIDA maps;
- Chordal networks;
- GLP topology generator.

We are also making experiments on “tiers” topologies, etc.

ALUSIM routing model allows unicast and multicast routing. It defines a routing protocol as a function $(L, R)route(r, l, m)$ where:

- m is the incoming message;
- r is the router the message m came from;
- l is the link the router r used to forward the message m ;
- L is a set of outgoing links where the message will be sent;
- R is a set of relay nodes to which the message will be sent. In practise all nodes that are destination nodes of links in L will receive the message but only those in R are allowed to relay it.

└ ALUSIM: a routing simulator based on MASCSIM

└ Packet-level simulation/event model

ALUSIM is a packet-level simulator developed atop MASCSIM.
As such, it features the two basic following events:

- Packet reception event;
- Packet emission event.

Packet emission

Definition

The packet emission event represents any emission of a packet from any node. It indicates which are the recipient nodes and which links have to be used.

The packet emission event will schedule a packet reception event at the current time + the estimated duration for transitting on the network link.

This duration is currently fixed but it should be computed according to the traffic on the link.

Packet reception

Definition

The packet reception event represents the reception of any packet on any node. It indicates which was the sender node and which link were used.

When a packet is received, the recipient node needs to find out to which neighbor nodes it will forward it. This decision is defined by the routing protocol. The reception event will then schedule as many emission events as selected neighbors.

Routing protocols

The following routing protocols have already been implemented:

- A set of routing schemes have been used to identify the needs and test the simulator.
- RIP;
- BGP (3 versions);
- NSR (Nicolas).

BGP - Initial implementation

- Node = Autonomous System = the unique router in this AS (Only EBGp simulation)
- Link = a physical link. A BGP connection will have the link as a first attribute
- The whole BGP peering finite state machine is implemented. 6 states: Idle, Connect, Active, OpenSent, OpenConform, Established,
- Security isn't targeted by simulation => No filtering policies

BGP - First optimization

- Leaving only 2 states for the BGP session: either Idle or Established;
- The new BGP session events needed only 5 events in spite of 15 in the old version;
- This has led to a BGP simulation which takes less time at the initializing phase.

BGP - Second optimization

A few issues remain:

- Still not enough — too much time before the first convergence → code profiling;
- looking up a destination = 70% .

Solutions:

- Invoking this method less often;
- Changing the data structure to make it faster;
- Deleting the forwarding table => loc-RIB in charge of forwarding.

Some results on the NSR routing protocol

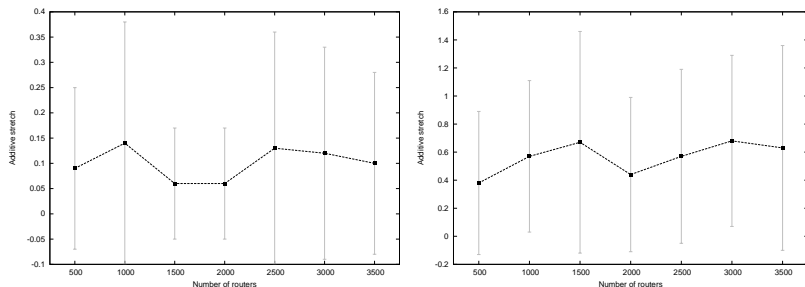


Figure: Measure of the additive stretch of NSR when applied to chordal (left) and GLP (right) topologies.

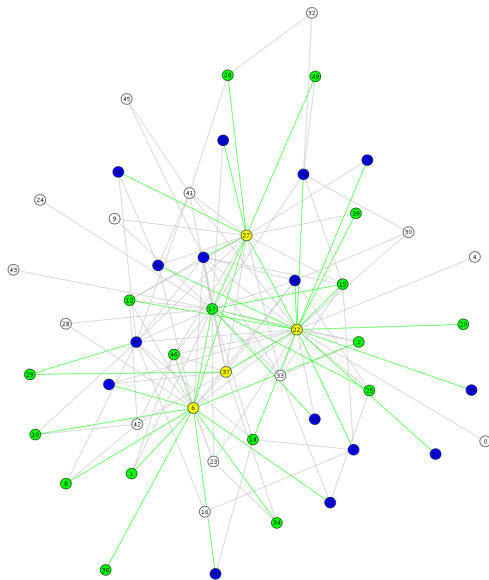


Figure: The aircraft view enables a precise monitoring of what is going on within the simulated network.

Implementation

ALUSIM has a polished OO model. The use of tricks (fast coding at the detriment of quality) is never accepted. The coding style follows the rules used in the Mascots project. As of June 2009, ALUSIM consists of:

- 132 classes;
- 7.764 lines of code (blank lines excluded);
- total of 23.838 lines of code in the repository.




ALUSIM allowed us to efficiently simulate simple schemes on 10.000 nodes. BGP was simulated on 3.000 nodes.

Software:

- introduce network dynamics (communication failures, technical maintenance);
- complete the implementation of the grid system;
- improve code quality/efficiency.

Publications:

- Recent submission to ReArch'09;
- Future submission to Simutools'10.

-  Hogie, L., Bouvry, P., Guinand, F., Danoy, G., and Alba, E. (2006).
Simulating Realistic Mobility Models for Large Heterogeneous MANETs.
In Demo proceeding of the 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'06). IEEE.
-  Jin, C., Chen, Q., and Jamin, S. (2000).
Inet: Internet topology generator.
Technical Report CSE-TR-433-00, University of Michigan at Ann Arbor.
-  Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001).
Brite: Universal topology generation from a user's perspective.
Technical report.



Y. Rekhter, T. L. and (Eds), S. H. (2006).

A border gateway protocol 4 (bgp-4).

RFC 4271.