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

Carry-le-Rouet

Mascotte, INRIA Sophia-Antipolis

Outline

What is our target?

We want to write a software-based simulation tool that enables the investigation of compact-routing protocols when applied to large networks (i.e. in the order of ten thousands nodes).
Let us call it Alusim.

A reuse of Madhoc

Alusim is not written from zero. It is built atop the framework of the Madhoc simulator.
Madhoc is a simulator for mobile wireless networks. It was developed in 2003-2007 and was used by 10 labs. We opt for reusing madhoc because:

- we know it;
- it proved efficient and stable;
- its code is "accessible";
- it is easy to extend/modify.

Applications of Madhoc

This lab/company	*did that*
ASTRA	VANETs
LABRI	Dynamic spanning tree
LITIS	Generation of dynamic graphs
VALORIA	Teaching, extensions
Lux	Broadcasting
Malaga	MOO
Toulouse	Mobile agents
...	...

Outline

On time-slicing simulations

Madhoc features a time-slicing simulation engine: time is discretized in constant steps. For the sake of flexibility, accuracy and performance, we want to have a discrete-event engine.

- simulation period is defined at conception time and cannot be changed afterwards;
- fixed period is a violent time approximation
- the existance of time slots when nothing happens degrades performance

Discrete-event simulation, a pragmatic approach

In discrete-event simulation, an event means: "it's going to happen *this* at *that time*".

this is what the event consists in, its code;

that time is the occurence date of the event.

The simulation starts with a number of initial events. When they execute, events schedule other events.

Outline

Discrete-event simulation in Alusim

Alusim is a packet-level simulator. 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 model

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.

Outline

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 - Full

- 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 $\rightarrow$ code profiling;
- looking up a destination = 70% .

Solutions:

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

BGP - NSR

Metrics

Alusim defines a number of metrics that dynamically provide information on the simulation:

- Stretch;
- Number of routing success;
- Number of routing failures;
- Number of entries in routing tables;
- Number of bits in routing tables.

Distributions of metrics are also available.

Implementation issues: should be harcode measure of resort to Aspect Oriented Programming?

Outline

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 ofcava*? *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.

Outline

The following tolopogy generators have been developed:

- Set of basic schemes (tree, grid, chain, random, etc)
- front-end of Inet (Inet must be installed);
- bridge to Brite;
- extraction of topologies from CAIDA maps.
- experiments on tiers topologies, etc.

Outline

Command line interface

Today, the simulation is meant to be invoked from the command line.

- It reads its configuration .alu files. A configuration file is a set of $key = \{a, b, c...\}$ entries.
- As output, it writes a log of the events as well as one (x, y) file per sensor, one (x, y) files for every pair of sensors, and a plot file for every (x, y) file.

The code

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 Mascopts 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.

The project is starting! Many things have to be done. Among them:

- define a trafic model;
- define model for network dynamics (failures, maintenance);
- improve user interface;
- enable simulation campaigns;
- define a simulation scheme (one to one, one to all, all to all?).