Status Report: *Marionnet*

“How to implement a virtual network laboratory in six months and be happy”

Jean-Vincent Loddo <loddo@lipn.univ-paris13.fr>
Luca Saiu <saiu@lipn.univ-paris13.fr>

Université Paris 13

Requirements

Teaching first- and second-year students to setup and configure networks.

Either:

- in the traditional way: in a room full of computers, wires, and devices

Or:

- In a simulated environment
  - faithful simulation
  - easy to use

We want it:

- Fast to write
- Robust

Let's give it a look

Live demo

Some data

Code size:

- OCaml ~12,000 lines
- C ~1,000 lines

Notable features:

- GUI (GTK+)
- Concurrency
- Fault tolerance
- File format backward- & foward-compatibility

6 man months

Reuse of C code (with modifications):

- UML, by Jeff Dike (User-Mode Linux)
- VDE, by Renzo Davoli

Emulation-level vs. *user-level* network

Treeviews

Nom	Type	Perte %	Duplication %	Bits inversés %	Retard min (ms)	Retard max (ms)
a						
eth0						
inward		0	0	0	0	0
outward		0	0	0	0	0
N1						
port0						
inward		0	0	0	0	0
outward		5	5	0.01	50	100
port1						
inward		0	0	0	0	0
outward		5	5	0.01	50	100
d1						
to a (eth0)		0	0	0	0	0
to N1 (port0)		0	0	0	0	0
b						
eth0						
inward		0	0	0	0	0
outward		0	0	0	0	0
d2						
to N1 (port1)		0	0	0	0	0
to b (eth0)		0	0	0	0	0

The 'a forest datatype

```
type 'a forest =
   Empty
 | NonEmpty of
     'a *                  (* first tree root *)
     ('a forest) *         (* first tree subtrees *)
     ('a forest);;         (* other trees *)
```

What could be a reasonable 'a?

The right data structure for treeviews...

...but for network graphs?

Messages and thunks

```
class ['a] queue :
  object
    val elements : 'a list ref
    val empty_condition : Condition.t
    val mutex : Mutex.t
    method private __empty : bool
    method dequeue : 'a
    method enqueue : 'a -> unit
    method prepend : 'a -> unit
  end
```

Our message-passing infrastructure is built on a `(unit -> unit) queue`.

More on thunks

```
let with_mutex mutex thunk =
  lock mutex;
  try
    let result = thunk () in
    unlock mutex;
    result
  with e -> begin
    unlock mutex;
    Printf.printf
      "%s raised in critical section.\n"
      (Printexc.to_string e);
    raise e;
  end;;
```

Passing an anonymous thunk to `with_mutex` is a pattern in our code.

What now

Marionnet will be distributed to students and used in courses about networks...

- on GNU/Linux live DVD systems

  “*Marionnettix*”, based on Knoppix

- by now only at the Université Paris 13 IUT

  We've ordered 1500 virgin DVDs for this semester

  so the thing should better work

We confide in a wider adoption in the future

About *Marionnet*

You're *welcome* to share and change *Marionnet*: it's free software, distributed under the GNU General Public License.

Thanks

http://www.marionnet.org

It will be online “Real Soon Now”
(yes, trust me)

Download it!

It's functional!!
(mostly)

It works!!!
(it should)

Ok, enough.

No questions yet?

This slide is over...
No more easter eggs...
This is the last one.