Distributed Systems

Eddy Caron[*]
ENS de Lyon

Contents

[*] hadriencroubois.com

Final grade 2/3 Final exam + 1/3 mid-term exam + 1/3 (project + partial)

We will use the programming language *Erlang* (1987 $\rightsquigarrow$ 1998)
• Concurrent, real time, distributed
$\Rightarrow$ Use BEAM

Variables can be integers, float, PID, functions, tuples, maps, ... and atoms (specific to Erlang).
There are built-in functions for message passing
Lists are not strongly typed, tuples are like python's
In Erlang, there is no overwriting of the variables (cannot affect them a new value).

1 Algorithm for Distributed System

1.1 Modelization

Definition 1 (Transition relation)**.** *Let $S = (\mathcal{C}, \rightarrow, \mathcal{I})$ be a transition system. An execution of S is a* maximal sequence $E = (\gamma_0, \gamma_1, ..., \gamma_n)$

Definition 2 (Terminal configuration)**.** *A* terminal configuration *is a configuration γ for which there is no δ such that $\gamma \rightarrow \delta$. Note that a sequence $E = (\gamma_0, \gamma_1, \gamma_2, ...)$*

$\rightarrow$ this notion is very tricky in parallel programming, as it is difficult to know whether all the executions are finished or not.

Definition 3. *A configuration δ is* reachable *from δ (notation $\delta \rightsquigarrow \gamma$), if there exist a sequence $\gamma = \gamma_0, \gamma_1, \gamma_2, ..., \gamma_k$ with $\gamma_i \rightarrow \gamma_{i+1}$ for all $0 \leq i < k$. Configuration δ is reachable if it is reachable from an initial configuration.*

Definition 4 (System with Asynchronous Message Passing)**.** *Cf APPD course*

1.2 Fairness

Definition 5. *An execution is* weakly fair *if no event is applicable in infinitely many consecutive configurations without occurring in the execution*

Definition 6. *An execution is* strongly fair *if no event is applicable in infinitely many configuration without occurring in the execution.*

1.3 Network topology

Many topology exists, including :

- Ring
- Tree
- Hypercube
- Star
- Clique

You can "change the topology" of your network to adapt and algorithm, for example taking a star sub-graph of a clique.

1.4 How to write a distributed algorithm?

- A specific code for a specific node (or family of node)
  - The sender code and the receiver code
  - The initial code and the non-initial code
- One code for all
  - The same code is executed on each node
  - A requirement can switch to the right code for a node

Use label to separated the code between initiators, non-initiator or reception of a specific message (for example stopping the execution).

Warning The execution flow is not clear: there is no assumption about the label selected for the execution (it is randomly chosen), so the algorithm must run for all the chosen labels for al the processors in the current state.

It is useful to try to find an incorrect workflow to test whether the algorithm is correct.

No assumptions must be made on the execution time (especially linking synchronization with the size of the code)!

2 Communication protocols

Example Write an algorithm to exchange informations between both process

```
vars:
$data_{in}$=N
$data_{recv} = -1$
is_sent = -1
$Label_1${is_sent==1}
begin
send $\langle$msg,$data_{in}\rangle$ is_sent=0
end
$Label_2${receive $\langle$msg,i$\rangle$}
begin
$data_{recv}$=i end
```