Modeling and Analysis of Wireless Sensor Networks (WSN)

VERIMAG
Olivier Bezet, Florence Maraninchi and Laurent Mounier

France Telecom R&D / VERIMAG
Ludovic Samper

Introduction to WSNs

- Huge networks : hundreds to thousands nodes
- No infrastructure
- Limited sensing, computation, and wireless communication capabilities
- Low cost, low power
- Applications : detection or monitoring an event in a distributed manner

Overview of the presentation

1. Related Works, research field in Sensor Networks

2. A case-study with GLONEMO : a sensor network model

3. Toward exhaustive verification

Existing Works, protocol layers

- Medium Access Control
- Routing
- Self-Organization

Need tools to evaluate this protocols

Existing Works, simulation

- Classical network simulators, not dedicated to sensor networks :
  - NS2 (The Network Simulator), Opnet, Glomosim, ...
  - NAB (Network in A Box)
- Sensor network simulators :
  - PowerTOSSIM, extends Tossim the simulator of tinyOS.
    In PowerTOSSIM, the consumption is computed from the number of transmission and from the number of instructions executed.
  - Avrora, written in Java and cycle-accurate
  - Atemu, executes binary code
  - ...

Related works : formal verification

A few (published) case-studies, with non-dedicated tools :

HyTech [2002] :
- verification of functional properties inside a single (TinyOS) node ;
- simulation at the network level (with very abstract nodes).

UPPAAL [2005] :
- specification of a MAC layer protocol ;
- verification of timed properties.

RT-Maude [2006] :
- specification of the OGDC algorithm (“auto-configuration”) ;
- verification of timed properties on a small network (6 nodes)

Our Approach

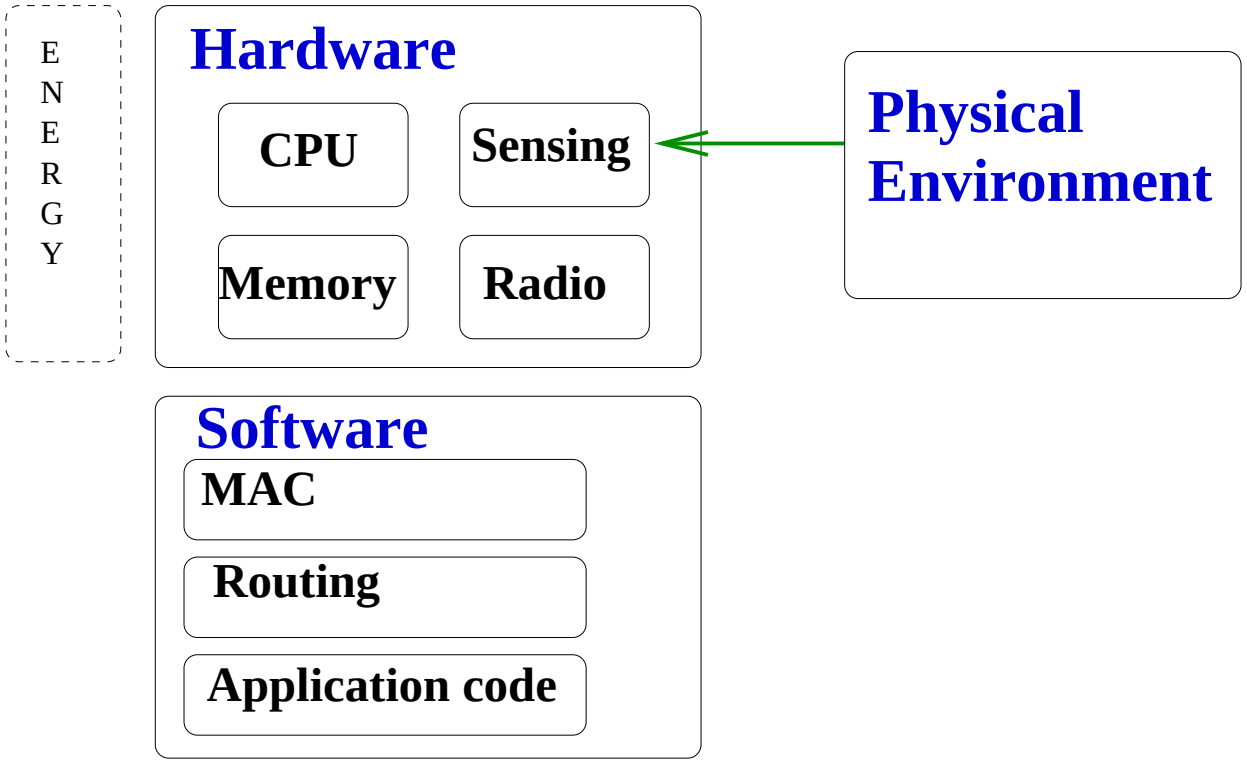
1. to define a global and accurate formal model of a WSN
   $\longrightarrow$ the “Glonemo” project

2. to experiment with existing verification tools (Lustre, IF)
   $\longrightarrow$ to find interesting properties to validate
   $\longrightarrow$ to understand the current limitations
   $\longrightarrow$ to propose the necessary extensions . . .

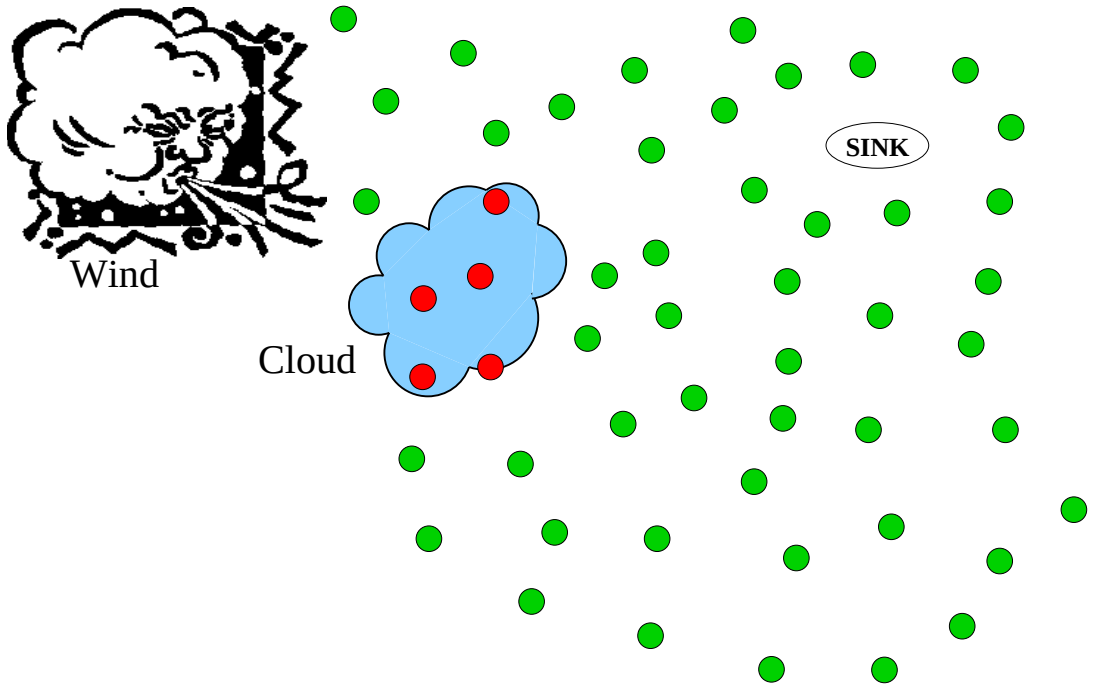
$\Rightarrow$ definition of sound abstraction relations :
- taking into account the energy consumption
- that can be applied in a component-wise fashion

GLONEMO : GLObal NEtwork MOdel

- **A global model**
  - Detailed Hardware
  - Software : the protocol layers and the application code.
  - Physical Environment

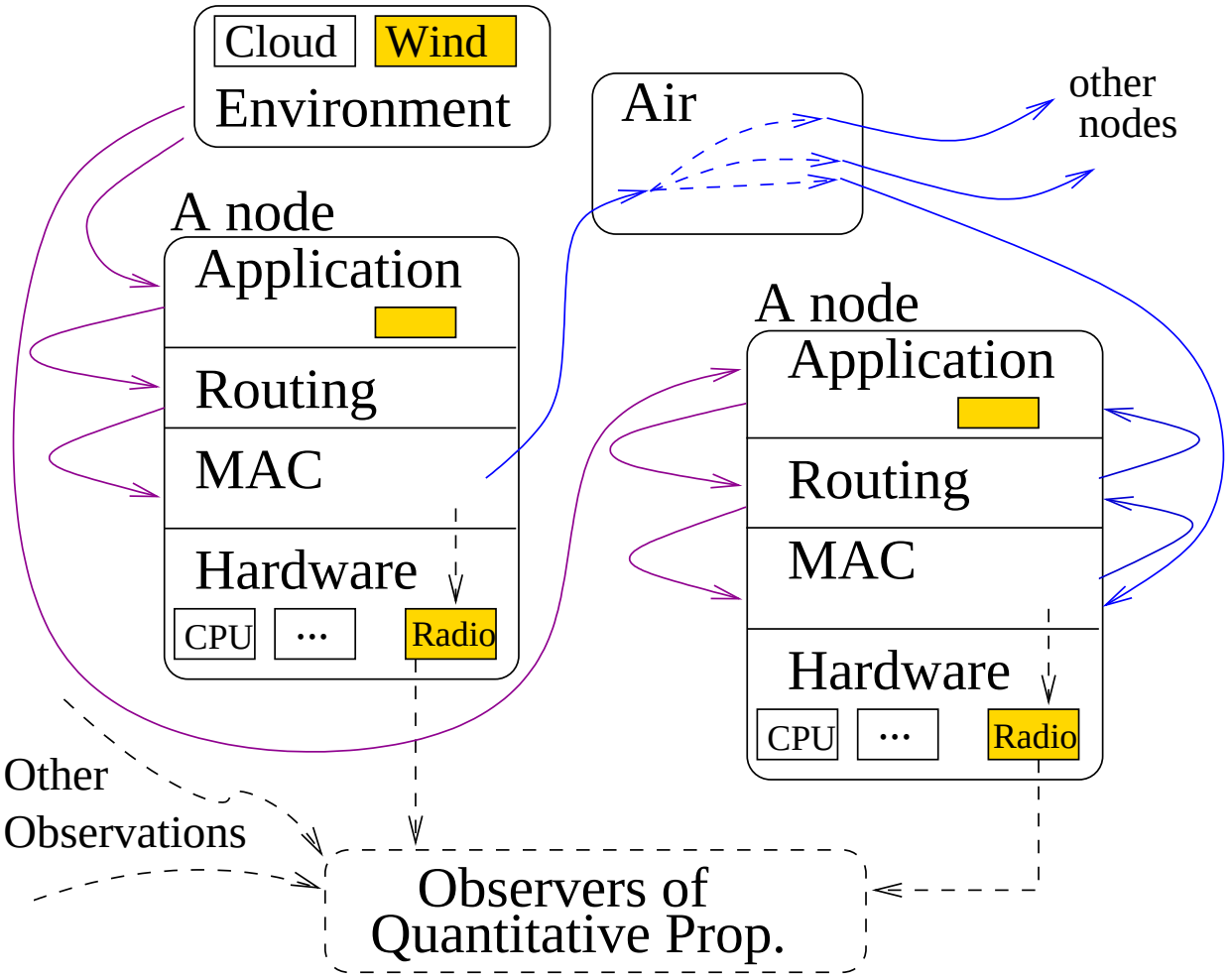


Typical Example, case-study



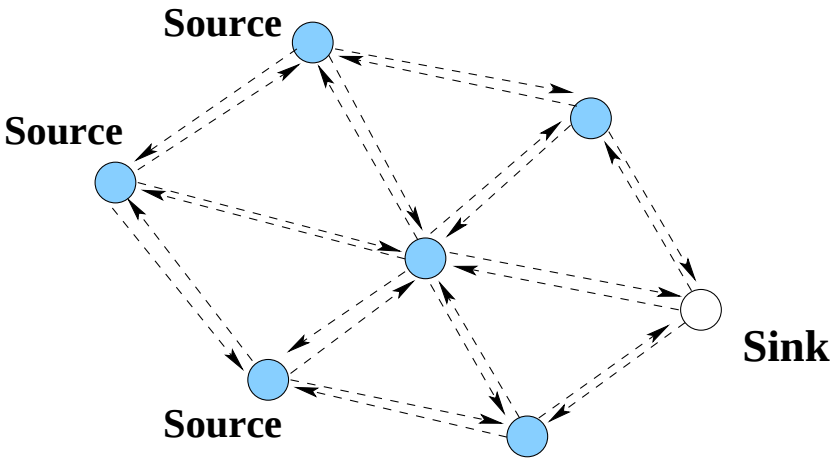
- **Application :** Detection of a radioactive cloud
- **Routing :** Directed diffusion
  (C. Intanagowiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva)
- **Medium Access Control :** A preamble sampling MAC protocol
- **Environment :** A cloud moving under the influence of the wind.

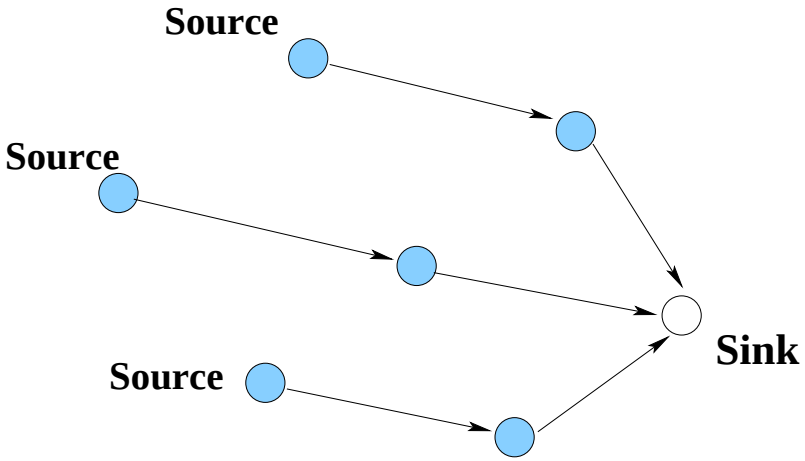
Structure of the model



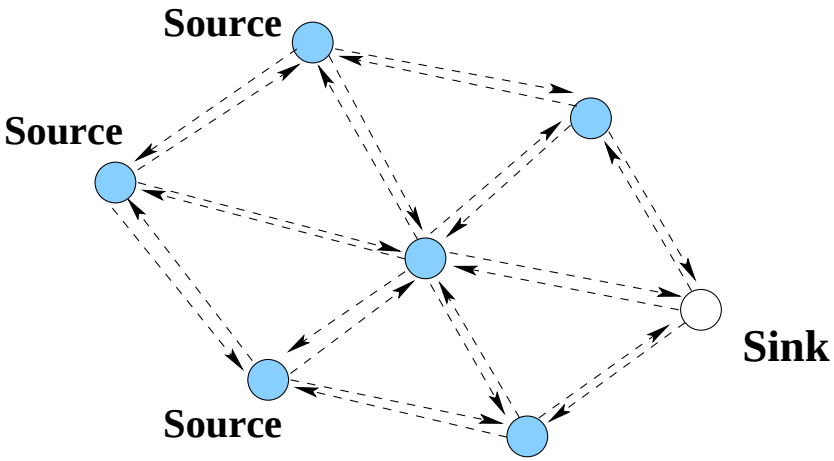
Parallel processes with synchronization

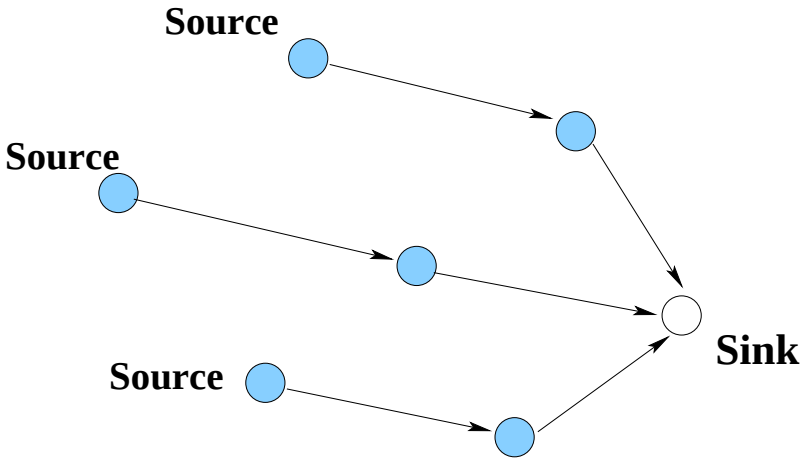
The Routing Protocol, Directed Diffusion

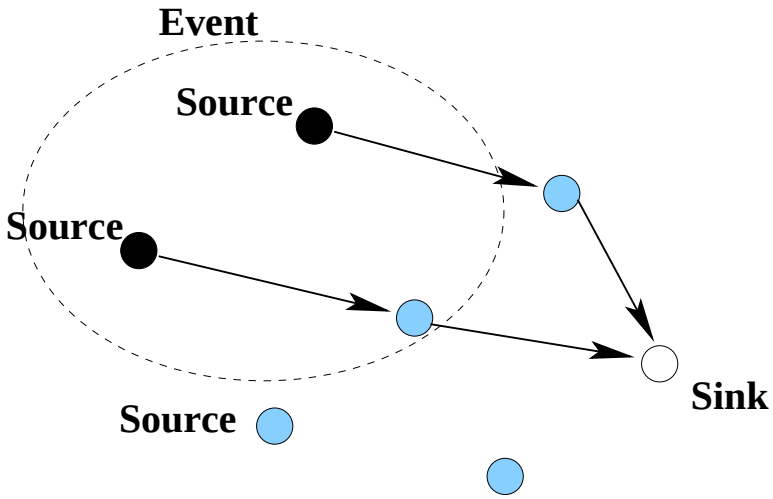




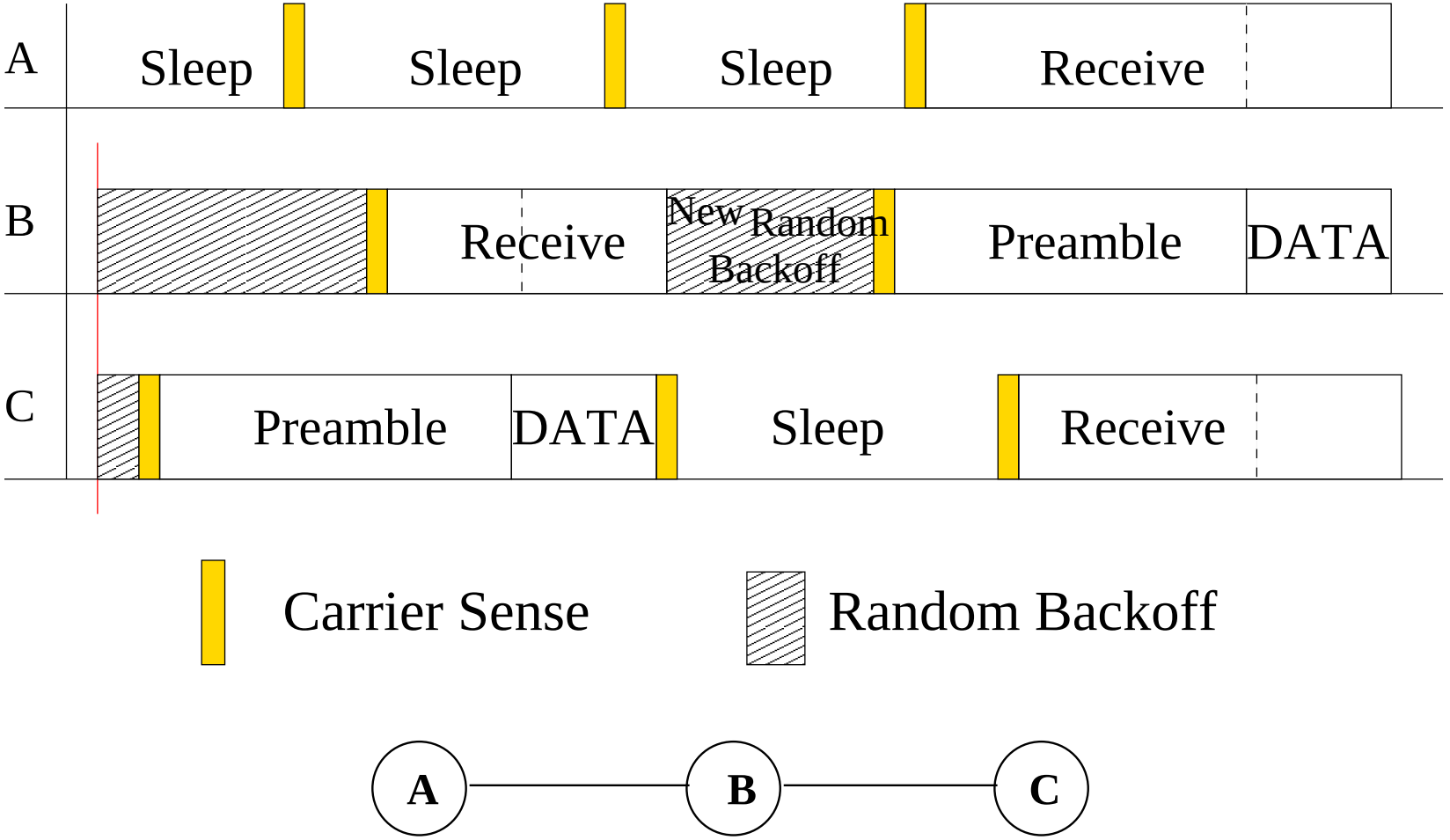
The Routing Protocol, Directed Diffusion





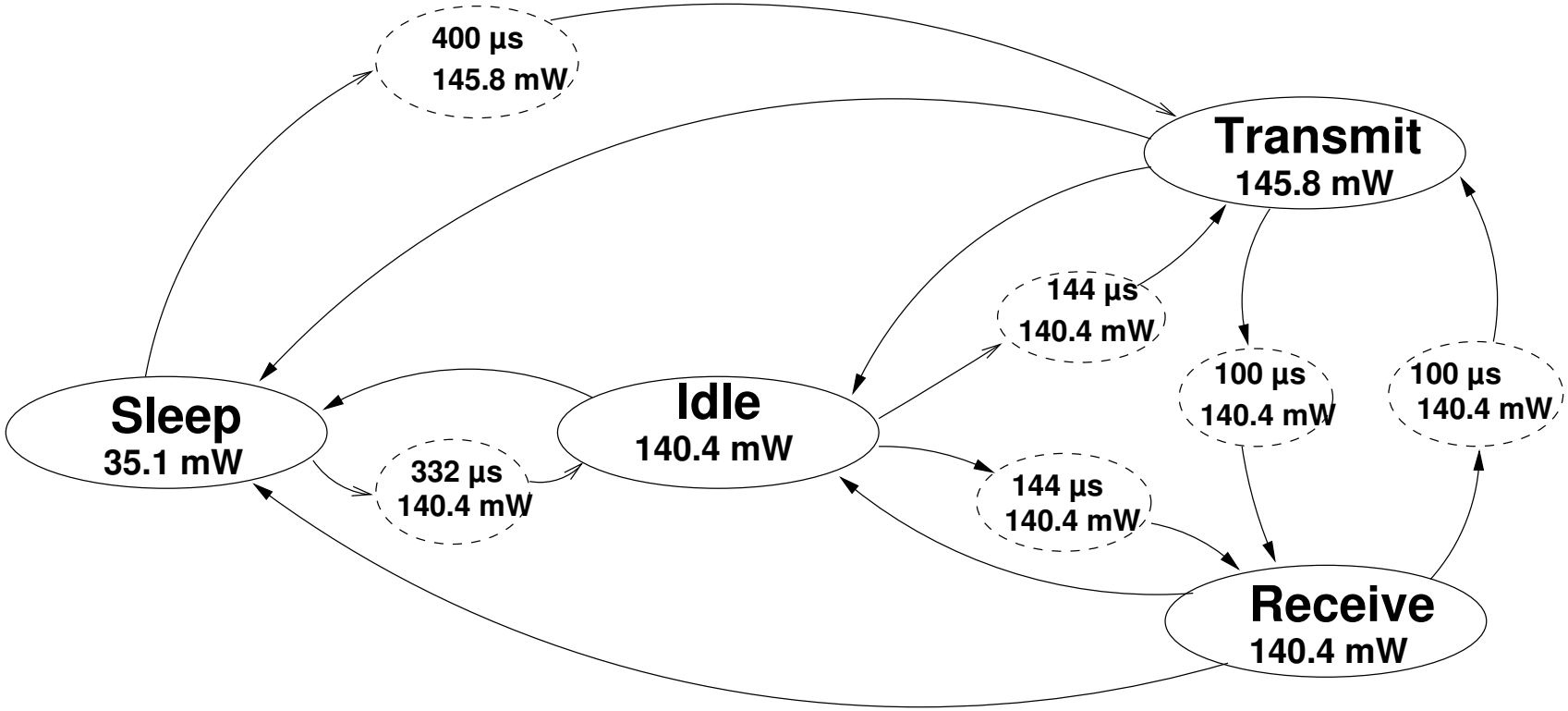


The Medium Access Control protocol, a preamble sampling MAC protocol :



The consumption model of the radio

The MAC layer drives this automaton.
An "observer" checks the current state to calculate the consumption of the node.



Values of the Motorola MC13192

Tools used to program the model

- REACTIVEML (Louis Mandel, LIP6) :
  - The ML-language with parallelism
  - As expressive as the Caml language
  - Parallelism is a top-level primitive
  - Synchronous Language

  The hardware model, the software and the simulation engine are implemented with REACTIVEML

- LUCKY (E. Jahier, P. Raymond, VERIMAG) :
  - A constraint-based language
  - A language for describing and simulating stochastic reactive systems
  - Lucky is connected to REACTIVEML

  The cloud and the wind are implemented with LUCKY

GLONEMO, conclusion

- An efficient simulator

- Realistic simulations thanks to the environment model :
  We have run simulations with this environment model and with classical Poisson processes to generate the packets, and the results where completely different.

- This implementation will help us to build sensor network models in other formalisms

Current and future works

Toward Exhaustive Verification :

1. A complete detailed model in LUSTRE

2. The problem of state space explosion :
   - Abstractions
   - Representation of cost automata

3. A simplified model in IF (not presented here)

From GLONEMO to LUSSENSOR

→ Apply the LUSTRE toolbox facilities to the GLONEMO model

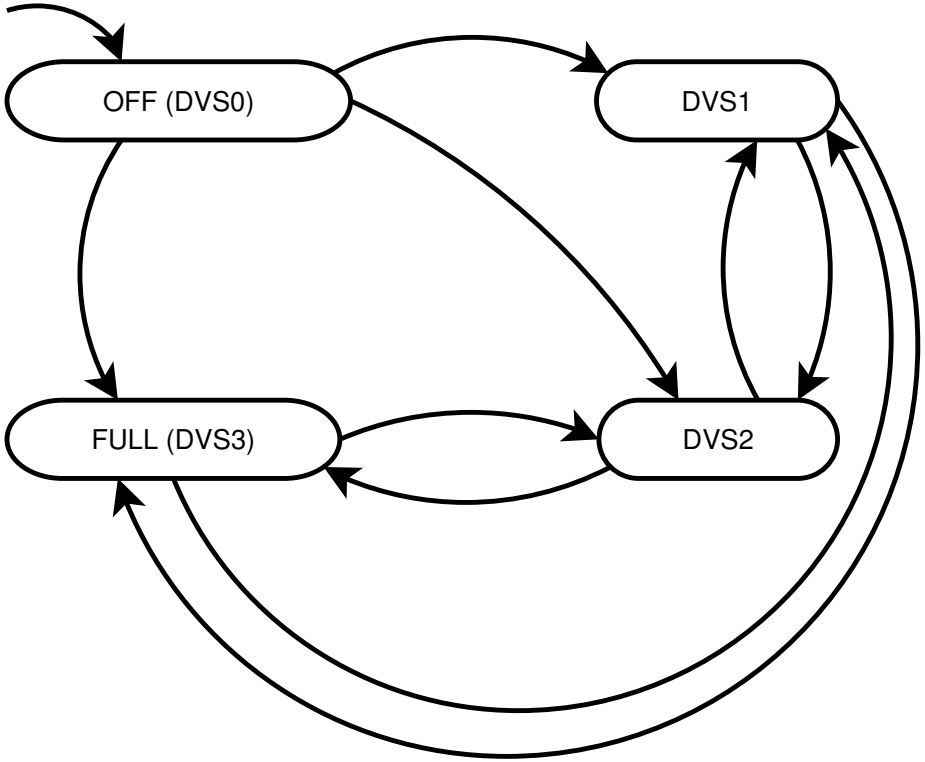
REACTIVEML to LUSTRE translation :

– both are synchronous languages
  → same semantics for time, parallelism
  → same computation model for energy consumption
– structural translation : RML processes → LUSTRE nodes

But

– unbounded data structures not allowed in LUSTRE
  ⇒ no dynamic node creation in the LUSTRE model
– channel modeling (collisions, preambles, data received)
  ⇒ based on matrices in the LUSTRE model

Example : The automaton of the CPU



CPU with DVS (part of the code)

```
let
  -- Assertions
  assert #(X0, X1, X2, X3) and (X0 or X1 or X2 or X3);
  assert #(mode[DVS0], mode[DVS1], mode[DVS2], mode[DVS3]);
  -- Manage CPU modes
  X0 = true  -> (pre(X0) and not(mode[DVS1]) and not(mode[DVS2])
                 and not(mode[DVS3])) or mode[DVS0];
  X1 = false -> (pre(X1) and not(mode[DVS0]) and not(mode[DVS2])
                and not(mode[DVS3])) or mode[DVS1];
  X2 = false -> (pre(X2) and not(mode[DVS0]) and not(mode[DVS1])
                and not(mode[DVS3])) or mode[DVS2];
  X3 = false -> (pre(X3) and not(mode[DVS0]) and not(mode[DVS1])
                and not(mode[DVS2])) or mode[DVS3];
  -- Calculate energy wasted
  energy = if      X0 then POWER0 * TIME_SCALE
           else if X1 then POWER1 * TIME_SCALE
           else if X2 then POWER2 * TIME_SCALE
           else            POWER3 * TIME_SCALE;
tel
```

LUSSENSOR : conclusion

LUSSENSOR is an accurate LUSTRE model of a WSN :

- all the layers have been implemented
- each consuming module is taken into account (RAM and flash memories, CPU, sensor, and radio)
- easily obtained from GLONEMO

$\Rightarrow$ needs to be “simplified” to go through verification

$\Rightarrow$ simplifications can be modular

$\Rightarrow$ well-defined abstraction techniques are necessary . . .

Toward Exhaustive Verification : Properties

Using Formal Methods, we want to bring information that are beyond the scope of simulations.

For instance, we would like to point out rare scenarios.

Interesting properties to verify :

- We want properties about the energy consumption.
  Example of such properties :
  - Maximum of energy spent during time t
  - Shortest lifetime of the network

- Those properties imply finite sequences, able to be verified in practice

Toward Exhaustive Verification : Abstractions

Abstractions to reduce the number of states

Examples

– Change the accurate model of consumption with a simpler one

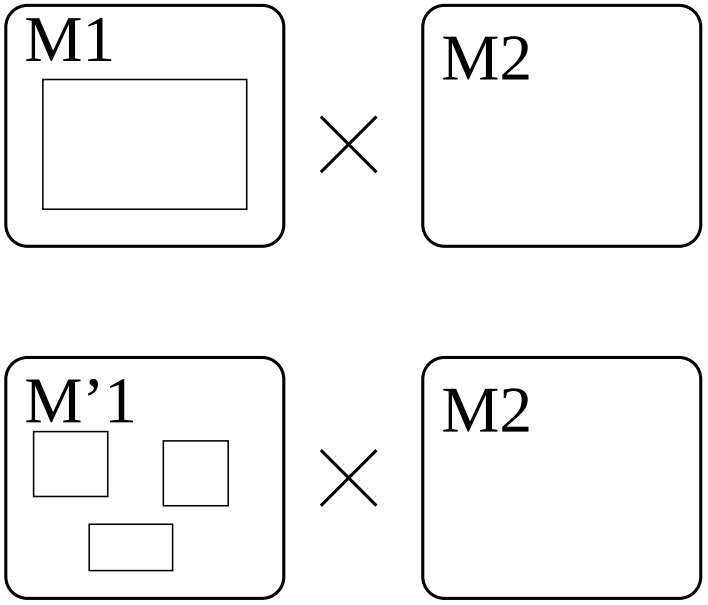
– Model accurately the consumption of one node and abstract the rest of the network

Abstractions should not be hidden in the model

Requirements for the Formalization

- Energy Models M_1, M_2, ... representing parallel activities running on the same source of energy or not
- Parallel Composition $\times$ of these machines, yielding an energy model of the parallel system
- A partial order on machines : $M_1 \preceq M_2$ if M_1 is a more precise model than M_2
- A pre-congruence property : if $M_1 \preceq M_2$ then, for any N, $M_1 \times N \preceq M_2 \times N$.
- All the operations of a language, the encapsulation operation for example : if $M_1 \preceq M_2$ then, encaps$(M_1) \preceq$ encaps(M_2).

Modular abstractions



M_1' is more precise than M_1.

The consumption evaluated with the M_1' model must be smaller than the one evaluated with M_1 ; and this relation must stay true after composition.
This is a worst-case framework hence more precise means smaller.

Definitions

$$M = (Q, Q_0 \in Q, I, O, T \in Q \times \mathcal{B}(I) \times 2^O \times Q, \mathcal{F} : Q \to \mathcal{L})$$

- Q, states
- Q_0, initial states
- I, Inputs
- O, Outputs
- T, Transitions
- $\mathcal{F}$, Cost Function
- $\mathcal{B}$, Boolean formulas on inputs I
- exec (M, I) = The trace of M with the inputs I (Projected on the costs : $\mathcal{L}_0\mathcal{L}_1\mathcal{L}_3\mathcal{L}_4...$)
- exec (M, I, k) = Traces of M with the inputs I of length k

$$M_1 \preceq M_2 \iff \forall I, \forall k \geq k_0, \int_{\text{time}} \text{exec}(M_1, I, k) \leqslant \int_{\text{time}} \text{exec}(M_2, I, k)$$

Where,

$$\int = \sum_i \mathcal{L}_i$$

Example

trace $(M_1) = a_1 a_2 a_3 \cdots a_k$

For example,

cost(trace$(M_1)) = \mathcal{L}(a_1) + \mathcal{L}(a_2) + \cdots + \mathcal{L}(a_k)$

Or

cost(trace$(M_1)) = max\{\mathcal{L}(a_1), \mathcal{L}(a_2), \cdots, \mathcal{L}(a_k)\}$

Synchronous Product

trace ($M_1 \times M) = (a_1, b_1)(a_2, b_2)(a_3, b_3) \cdots (a_k, b_k)$

cost(trace$(M_1 \times M)) = (\mathcal{L}(a_1) \bigoplus \mathcal{L}(b_1)) + \cdots + (\mathcal{L}(a_k) \bigoplus \mathcal{L}(b_k))$

Where,

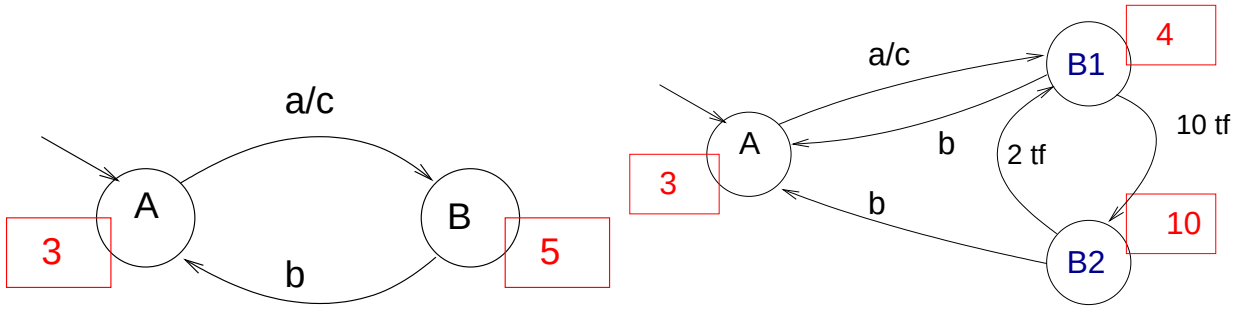
$\bigoplus = max$ if both M and M_1 are on different batteries

$\bigoplus = +$ if M and M_1 are on the same battery

Example

– The functional behavior of 2 is more detailed than that of 1 : functionally, the state B has been split into B1 and B2. Automaton 2 simulates automaton 1.

– The evaluation of the energy spent is smaller : the proportion 10/12 of the time is spent in a state that costs 4, and 2/12 of the time in a state that costs 10. For periods long enough, being in states B1 or B2, the energy consumed is less than the energy consumed in state B.



$$\frac{10\times 4+2\times 10}{12} \leq 5.$$

We can use tools like NBac, ASPIC or FAST to validate such abstractions

Outline

1. GLONEMO
2. LUSSENSOR, A Sensor Network model in LUSTRE
3. – Formalization abstractions
   – Representation of priced automata

State space representation

In the context automata with clocks and energy,

$$\text{a state} = \text{control} \times \text{clocks} \times \text{energy}$$

We need also a symbolic representation of the state space.

We need a representation for :

- Control, this set is already finite, thus it is ok.
- Clocks
- Energy

State space representation

Existing tracks :

- If clocks and energy are discrete, their representation is finite (on finite traces) but it can be huge.
- For timed automata, there exists symbolic representations.
- People have extended the representation of timed automata for cost automata
- **But** their representation is specific for their problem and it does not fit our needs :
  - They need to find the min cost scheduling of a program, (“shortest” execution path
  - Regarding our properties, we are more interested in the longest path.

→ Current work to find such a representation

Outline

1. GLONEMO
2. LUSSENSOR, A Sensor Network model in LUSTRE
3. – Formalization abstractions
   – Representation of priced automata
4. Conclusion

Conclusion

- GLONEMO is an efficient simulator written in REACTIVEML
- LUSSENSOR is almost the same model in LUSTRE
- LUSSENSOR has to be simplified in order to use the LUSTRE toolbox
- Those properties involve time and energy, we define what is a correct abstraction with respect to the "Modular Worst-Case Energy Consumed" principle
- We will use/extend the LPTA formalism to have an efficient representation in order to find the worst case (energy) execution. A new tool dedicated to those kind of properties ?

Questions ?