On the Importance of Modeling the Environment when Analyzing Sensor Networks

Ludovic Samper
France Télécom R&D
Email: Ludovic.Samper@orange-ft.com

Florence Maraninchi,
Laurent Mounier
VERIMAG
2, av. de Vignate, F38610
Email: Florence.Maraninchi@imag.fr,
Laurent.Mounier@imag.fr

Erwan Jahier,
Pascal Raymond
VERIMAG
Email: Erwan.Jahier@imag.fr,
Pascal.Raymond@imag.fr

***Abstract*— A sensor network may be considered as a large and complex computer system embedded in the physical environment that has some influence on the sensors. The environment is the source of (almost) all activity that occurs in the network. With a simple example modeled in our tool GLONEMO, we show the influence of an environment model that allows to describe correlated stimuli on the set of sensors at a given instant, and also correlations between successive instants.**

I. INTRODUCTION

Apart from its functional behavior, the main characteristic of a sensor network is power consumption. All the elements of a network have some influence on power consumption: the hardware of the nodes, the method used to access the radio functionalities, the communication protocols that determine how long the radio should be on for a particular node, the application, and even the environment of the network, that stimulates the sensors and is often the source of the main activity in the network. Power consumption has to be estimated in advance, and this can be done by simulating a model.

Recently, we proposed the tool GLONEMO for the accurate and *global* modeling of sensor networks. See [9]. One key feature of GLONEMO is the possibility to include a model of the physical environment that influences the sensors. We performed some experiments, modeling simple ad-hoc networks, and we observed that the environment has a strong influence on the behavior of the system.

In this paper, we illustrate this fact by comparing the behaviors of the same network (same hardware, same protocols, same application) when simulated with two environment models: the first one is a model of independent stimuli on all the sensors, both spatially and temporally, that could be used in a traditional network simulator where Poisson laws are used to abstract the arrival of packets at each node; the second one is an operational non-deterministic model of a moving cloud, that generates correlated stimuli on the sensors, both spatially and temporally.

We think that the crucial point is to be able to describe correlated stimuli.

The structure of the paper is as follows: in Section II, we briefly describe the elements of the global model we use; Section III concentrates on the part of the global model that describes the physical environment, and explains how to compare simulations. Section IV mentions some related work, and Section V concludes.

II. THE GLOBAL MODEL ELEMENTS

The global model described in [9] includes an accurate modeling of the energy consumption. It is made of the following elements: the hardware, the protocol layers, the application, and the environment. We describe the application, the protocols and the hardware we use in our experiments about the model of the environment.

Modeling the energy consumed in all these elements provides an accurate estimation of the lifetime of the sensors.

A. The scenario and the Application

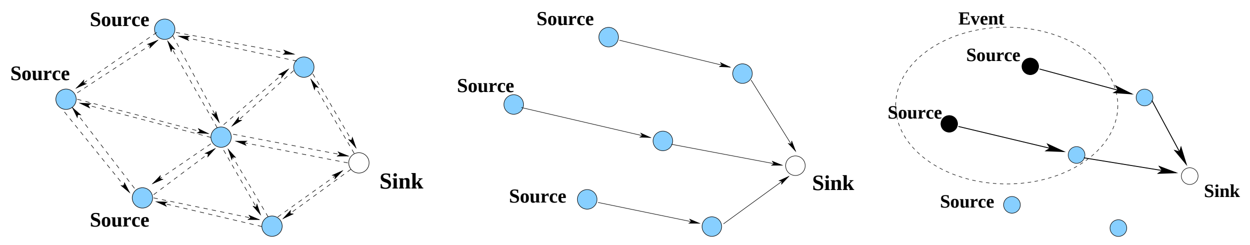
To point out the importance of the environment, we choose a monitoring application that really depends on the environment.

The role of our network is to warn the sink that there is a danger. This danger can be the presence of a radioactive cloud, or the presence of an enemy in a military context, or any other moving “object”. In the sequel we consider a cloud. When the danger is detected by a node, it sends an alarm packet to the sink. Because the cloud will stay some time on the same node, the sensor could send duplicated alarm messages. Hence, to avoid the emission of multiple useless messages, we implement data processing on sensors: the alarm packet is sent only once for each detection of the cloud (the edge between ”no cloud” and ”cloud”). If the cloud does not move, then the nodes under the cloud will send only one alarm packet.

To simulate the network, a traffic pattern is needed. For this example, one solution is to model the packet arrival at each node, and another solution is to describe the environment as an additional process.

B. Protocol layers: Routing

In our application, the need to communicate comes from the environment. We implement directed diffusion [5] which suits this application perfectly. Figure 1 illustrates the three steps of the routing protocol. The sink broadcasts an interest message to the whole network, fig 1(a). This message is sent using a flooding routing mechanism. Each node retransmits all the



(a) Broadcast interest (b) Setup gradient (c) Send alarm

Fig. 1. The three nodes on the left are concerned by the interest request. All the nodes set up gradients. And only the nodes which detect the Event and which are concerned by the interest will send an alarm according to the gradient path.

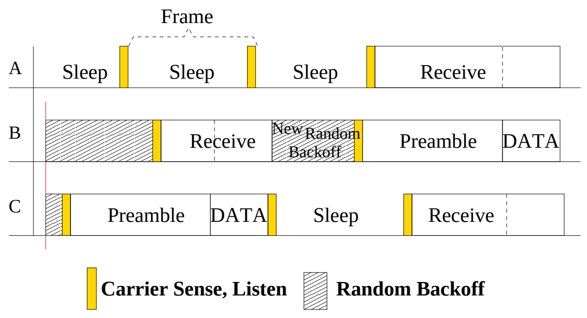


Fig. 2. Medium Access Control with back-off: B and C send messages. A and C are not in the range of each other.

packets it receives except the ones it has already forwarded, hence preventing packets from doing loops in the network. An interest concerns a specific task and also some nodes, here the three nodes on the left are concerned. When a node receives an interest, first it checks whether it is concerned by the request. Then it sets up a gradient between the sender and itself (fig 1(b)) and it forwards the packet. The task can be a periodic one, something like ”send the temperature once an hour” but it can also depend on the sensing: ”if the temperature increases sharply, send a message”. Once again, we want to point out the role of the environment, so we use the second kind of interests. Otherwise, the environment would only change the content of the messages but not their number nor their frequency. The nodes concerned by the request have to send an interest response. This packet will be routed according to the interest gradients. The response will take the route taken by the incoming request. We can already notice that all the messages from one sensor to the sink will always be routed through the same nodes. Here, the whole network is concerned by the interests and only the nodes that detect the event (the cloud for example) will reply. On figure 1(c), only two nodes have detected the event.

C. Protocol Layers: Medium Access Control

For observing the effect of the environment on a sensor network, the MAC layer cannot be omitted.

A perfect MAC layer (with no collisions and no delays) is sometimes used for simulations. This assumption is too strong when there is an intrinsic cause of collisions. In our case, collisions can occur when the sink flows an interest request through the network but also when the network reacts to an event in the environment, hence we cannot assume the channel to be collision free. On the contrary, we want to observe how the collisions influence the behavior of the network.

Another option often used for simulations is a IEEE 802.11 MAC protocol that has been designed to avoid most of the collisions (some collisions still occur, however, because of the hidden nodes problem). We cannot use this option either. Indeed, our global model includes an accurate modeling of the energy consumption; since the radio is the most important source of consumption on a sensor, it is far from realistic to build a global model around a 802.11 MAC protocol which is not optimized to be energy-efficient. We need to include the model of an energy-aware MAC protocol.

The Medium Access Control protocol used for this experiment is a typical sensor MAC protocol. It is a preamble MAC protocol like Low Power Listening [8] or WiseMAC [3]. See figure 2. Nodes periodically check whether the channel is free. If the channel is busy, the node will let its radio on to get the packet that follows the preamble. Otherwise, it goes back to sleep. We call a complete cycle of the carrier sense period and sleep period a *frame*. The *duty cycle* is the ratio between the length of the listen period and the length of the frame. To avoid collisions we implemented a back-off. The sender has to wait for a random time before emitting anything, then it scans the channel and if the channel is clear (Clear Channel Assessment, CCA), it sends the preamble and then the message. Otherwise, it delays the emission by setting a timer at random between 0 and cw_{max}. A preamble precedes each data packet for alerting the receiving node. All nodes in the network sample the medium with a common period. Figure 2 gives a possible timing behavior for three nodes A, B and C.

D. The Hardware

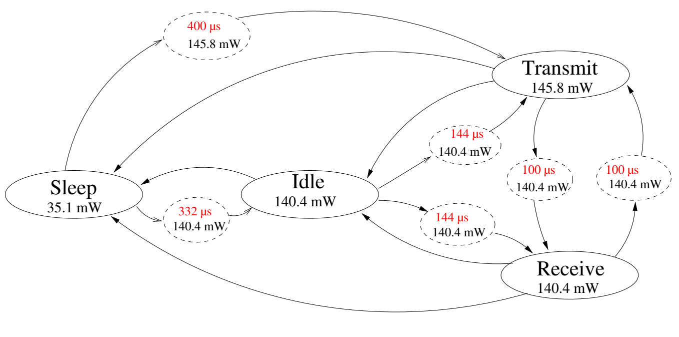


Fig. 3. Energy consumption of the radio

For the hardware parts, we may model the consumption of the CPU, the memory, or any other hardware part that consumes energy.

The idea behind the formal models is the following: each consuming part of the hardware is described by an automaton whose states are labelled with powers. Figure 3 corresponds to the radio. A particular execution of the global model (this automaton synchronized with other ones) will correspond to some path in this automaton, for which we can compute the total energy, knowing how much time we spend in each state.

E. Two Environment Models

Modeling the traffic in a network means modeling the packet arrival at each terminal. Modeling the packet arrival at each node is an issue that has been extensively studied in classical networks. See for example [4]. A classical abstraction is to use a Poisson model. Interarrival times are exponentially distributed with rate parameter λ: $P(A_n \leq t) = 1 - exp(-\lambda t)$.

Another option is to run the network and its environment together. Instead of modeling the packet arrival, we model the environment as a process, directly. This usually requires a more precise specification of the environment. In a sensor network dedicated to achieve a certain service, this more precise specification is often available. In other networks, like the web, the environment is made of all the users. The chaos model (anything can happen, at any time) may be sufficient.

We describe the two alternative environment models for our application below.

1) Poisson Processes: In sensor networks, all the nodes in the network have the same role except the sink. Of course, because some nodes forward packets from or to the sink they may achieve higher throughput. However, this does not concern the environment modeling nor the packet arrival rate at each node. Hence, the rate parameter λ will be the same for each node. This corresponds to the fact that we assume the cloud can appear anywhere on the sensor field with the same probability.

In this model, the danger detection events at one node are built with a local timer set using an exponential law, which means: the danger events (hence the alarm packets) at one node are independent of the past; the danger events (hence the alarm packets) at two neighboring nodes are completely independent, too.

The law to generate the next packet arrival is:

$$A_n = \lfloor -\lambda \times log(x) \rfloor \quad (1)$$

where x is a random number generated by a uniform law on $[0, 1]$. Our global model is discrete, and the environment model has to be discrete too. That is why we have to take the floor part of the generated number.

2) Modeling the environment with Lucky: The Lucky [6] language proposes to describe the behavior of a dynamic system by a set of characteristic *variables* and a set of *constraints* over these variables. The constraints may relate the values of several variables at some point in time, but they can also be used to relate the values of variables at different points in time. When modeling the moving cloud, this means we can impose spatial and temporal correlations between the danger events at each node.

```
-----------------wind.luc---------------------
inputs { }
outputs {
    Wind_x : float ~min -5.0 ~max 5.0 ~init 0.0;
    Wind_y : float ~min -5.0 ~max 5.0 ~init 0.0;
}
locals { }
nodes {
  init : stable;
}
start_node { init }
transitions {
 init -> init ~cond
    abs (Wind_y - pre Wind_y) < 5.0 and
    abs (Wind_x - pre Wind_x) < 5.0
}
-----------------cloud.luc--------------------
inputs {
  Wind_x : float ~init 0.0;
  Wind_y : float ~init 0.0;
}
outputs {
  x_cloud: float ~init 400.0
                 ~max 1000.0 ~min -100.0;
  y_cloud: float ~init 300.0
                 ~max 1000.0 ~min -100.0;
}
locals { }
nodes {
  init : stable;
}
start_node { init }
transitions {
 init -> init ~cond
  (if Wind_y >= 0.0
   then ((y_cloud - pre y_cloud) >= 0.0
    and  (y_cloud - pre y_cloud) <=  Wind_y)
   else ((y_cloud - pre y_cloud) <= 0.0
    and  (y_cloud - pre y_cloud) >=  Wind_y))
 and
  (if Wind_x >= 0.0
   then ((x_cloud - pre x_cloud) <= pre Wind_x)
     and (x_cloud - pre x_cloud) >= 0.0)
   else ((x_cloud - pre x_cloud) <= 0.0
     and (x_cloud - pre x_cloud) >= pre Wind_x)
}
```

Fig. 4. Lucky programs for the environment

A Lucky program is intrinsically non deterministic. The Lucky execution engine is based on a constraint solver, for Boolean and numerical constraints. Executing a Lucky program produces a sequence of random values that respect the constraints. Lucky is connected to GLONEMO.

We model the cloud as a process interacting with the sensors, and describing the moves of a disk-shaped cloud according to the wind direction and the wind speed. Figure 4 is the Lucky program. The variables `Wind_x` and `Wind_y` represent a two-dimensional wind, which does not vary a lot. The cloud is a disk whose center has the coordinates `x_cloud` and `y_cloud`. The constraints may involve expressions like `pre x_cloud`, to talk about the previous

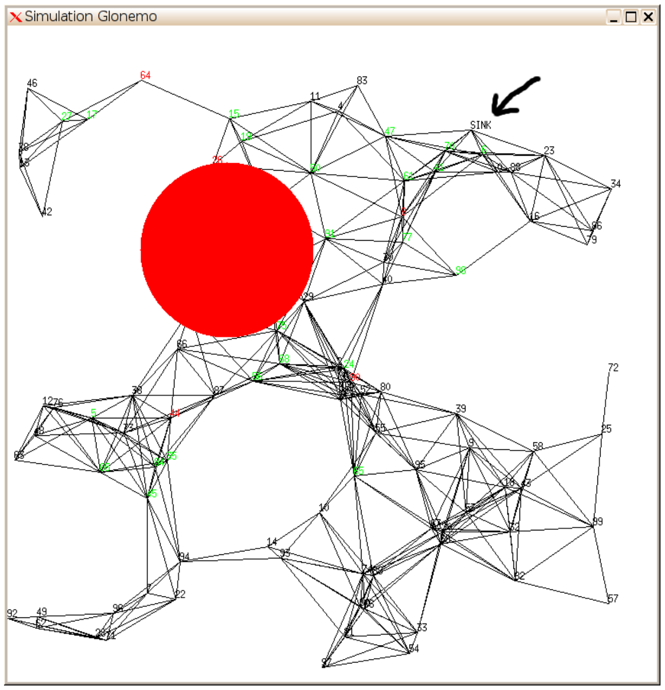


Fig. 5. Picture of a Glonemo simulation with the Lucky cloud.

value of the `x_cloud` variable. This allows to express constraints on sequences. The effect of the wind on the cloud is described by constraints of the form: `if Wind_x >= 0.0 then ((x_cloud-pre x_cloud) <= Wind_x and (x_cloud-pre x_cloud) >= 0.0)`. Generating a sequence of values for the tuple `(x_cloud, y_cloud)` could give: `(395.49, 385.98), (395.86, 386.15), (396.22, 386.33), ....`

Figure 5 is a picture of a GLONEMO simulation that includes the model of the moving cloud (the red disk). The black arrows shows on the Sink.

III. COMPARING SIMULATIONS

A. Conditions for Comparing Simulations

We run simulations for two identical networks except their communication patterns. The Lucky cloud activates the first, whereas Poisson processes are used for the second.

In order to observe the influence of the environment model, we need to compare Lucky simulations and Poisson simulations. And for this comparison to be meaningful, we must choose a criterion. This is not an easy task.

One idea is to compare simulations of the two models that send the same number of alarms during their lifetime. Note that the alarms in the Lucky simulation will be more dense, because the cloud activates several nodes at once.

To ensure this comparison criterion, we tune the parameter of the exponential law (λ is set to $1000 \times$ length of the frame, which is an indirect way of expressing a number of instants), and we write special constraints in the Lucky model in order to constrain the time during which the cloud is present. Technically, this is made possible by our Lucky model of the cloud, that allows to move it outside of the network, or to stop it, or to make it vanish.

B. Observations

We run the simulations with 14 mWh batteries for each node except for the sink. The simulation ends when the whole network is dead. The other parameters are:

- battery capacities per node = 50 J (13.9 mWh)
- 100 J for the sink
- 100 nodes (1 sink and 99 sensors)
- the nodes are on a 700 units square
- transmission range = 120 units
- length of 1 frame = 1 second
- duty cycle = 3%
- $\text{cw}_{\text{max}} = 100$

To evaluate the usefulness of networks, a good parameter is the ratio between the number of packets received at the sink and the total number of alarms emitted. For this experiment $\lambda = 1000 \times (\text{length of the frame})$ and the cloud just crosses the network. The results are summarized on the next table. It appears that the network works quite well when it is activated by Poisson processes. The rate of alarms packets received at the Sink is greater than sixty percent. With our modeling of the cloud, the rate falls down. Only ten percent of the alarms are received at the sink!

	Poisson processes	Lucky cloud
alarms emitted	76	210
alarms received	48	15
corresponding rate	0.63	0.07
number of collision:		
interest propagation	189	189
total	218	736
data only	29	547

If the rate is not hundred percents, it is because of the collisions. Let us observe where the collisions happen. On figures 6, 7, 8 and 9, we have taken pictures of running simulations. The cloud can be confined in the down-right corner (fig 7), or can go everywhere (fig 8) or everywhere except on the Sink (fig 9). The circles around a node represent the number of collisions for this node. It may have from 0 to 4 circles if it suffers between 0 to more than 40 collisions. When a node runs out of battery, a black disk (instead of its node number) represents it.

On figure 6, nodes are activated by Poisson processes. For this Poisson experiment, we increase the rate of packet arrival at each node to $\lambda = 92 \times$ length of the frame. i.e., on average, one alarm each $92\times$length of the frame. Here, most of the collisions occur on the route that goes to the sink. We can observe a funnel effect near the Sink. The nodes on that route are the first to die.

On the contrary, if the nodes are activated by a Lucky cloud, most of the collisions occur where the cloud went through. The huge number of collisions in Lucky simulations are local collisions. Observe figure 7: a lot of packets collided in the corner where the cloud went. Collisions happen because many neighboring nodes are activated at the same moment. Those nodes, in that corner, die first. On the opposite, nodes near the sink die later because they do not have a lot of packets to route.

	Poisson processes	Lucky cloud
alarms emitted	77	148
alarms received	44	15
corresponding rate	0.57	0.10
first node dead	67676	83775
all the nodes are dead	92790	93839
disconnected	84681	85924

The previous table shows information about the life span of our network in those two environments. The line *disconnected* means that some nodes are no longer connected to the sink. We show this parameter because as soon as the graph is unconnected, the network needs maintenance. It appears that even if the cloud makes the network send a little bit more alarms, the sensors will die later. In fact, with the Lucky cloud, the network works so badly that the sensors do not even use their energy. Alarm packets collide quickly after being sent whereas in Poisson processes, a single alarm involves many nodes and make them use their batteries.

We pointed out the behavior differences of the same network activated with or without an accurate model of the environment. We showed only some comparisons on the efficiency of the network, the lifetime and the number of collisions. However, we could have shown many other parameters. Indeed, GLONEMO permits to do so. The differences observed are so considerable that the study of precise environments is worth continuing.

IV. Related Work

Sridharan et al [11] after a survey on network simulators and on environment simulators propose to link a Matlab environment simulator with the sensor network simulator TOSSIM. The example taken is the monitoring of the health of a building structure. This is modeled as a state-space system which follows a differential equation. Matlab suits for that application because it is dedicated to solve differential equations.

SensorSim [7] is a discrete event simulator dedicated to sensor networks. In this simulator, the environment is not modeled but it is taken into account. Each node has a “Sensor protocol stack” that gets messages from a “Sensor Channel”. An analogy between the sensor channel and the wireless channel is done. This only difference is the propagation characteristics.

J-Sim [10] includes a Sensor channel and two propagation models are implemented for the phenomena: Seismic and Acoustic. The sensing phenomenon is created by the Target node which periodically generates stimuli that propagate on the sensor channel according to the propagation model.

Downard, in [2], extends the ns-2 framework to include support for sensor networks. Here also there is an analogy between the sensor channel and the wireless channel. But, the author goes further in that analogy. A new “radio” channel is created for each phenomenon. There are two types of nodes: the phenomenon nodes can ”communicate” only on the phenomenon channel and the sensor nodes can receive information about the phenomena and they communicate on the “real” radio channel. The phenomenon broadcasts its presence periodically by sending packets on the phenomenon channel. In order to avoid collisions between phenomena which would not be realistic, the mac channel used in the phenomenon channel is “basic”. Using a radio propagation model to simulate anything other than electromagnetic wave propagation is probably unrealistic, but above all it might be really expensive. Indeed, a routing layer, a mac layer and several nodes are involved for the modeling of a single phenomenon.

Demirkol et al [1] have motivations very close to ours. They propose a traffic model for sensor networks. Their model is a description of a target moving on a sensor field. The goal of the network is to detect the target. With the speed of the target, the density of the sensors, their sensing range and the sensing interval, they suggest an algorithm to generate realistic packet traffic. Basically, the idea is that if the target is somewhere at time t then it cannot be anywhere some moments later.

Even if the environment is not always included in sensor networks simulations, people are already conscious of the problems induced by this environment. At the routing layer, for example, new kinds of communications have been proposed in the literature, like gossiping and data gathering. With gossiping, the nodes exchange information locally and then send only one message to the sink. Data gathering consists in gathering information before sending it to the sink. This feature prevents redundant information from being sent to the sink. Those routing mechanisms are intended to be more adapted to sensor networks.

Our environment model allows to validate the assumptions that led to the design of these new communication schemes. For instance, in our example, gossiping is probably a good solution to avoid the emission of multiple redundant messages. Data gathering, however, does not seem to fit the problem. It reduces the traffic load near the sink, but this has little influence on the local traffic. These intuitions could be validated by simulating a global model that includes a precise model of the environment.

V. Conclusion

We illustrated with a small example the influence of a precise model of the environment. In fact, in all network simulations, there is a model of the environment. But in traditional network simulators, it is very abstract. We claim that these very abstract models, which are limited to the description of temporally and spatially independent stimuli on the sensors, are inadequate for sensor networks. Even if we have a precise specification of the global environment, and want to include it in the global model, it may still be the case that this can be done by some distributed algorithm, i.e. by some additional code on each node. However, in the general case, the model of the environment needs to be global. We describe it as an additional process, that interacts with the models of the nodes.

In our tool Glonemo, it is possible to describe the environment in a wide variety of languages and programming styles, ranging from pure deterministic algorithms to highly non deterministic sets of constraints. We think that this should be sufficient for the modeling of sensor networks. We will continue to explore the interest of these models, especially with the communication patterns that are emerging for sensor networks.

Finally, we considered *sensor* networks only, for the moment, and the model of the environment is used to evaluate non functional properties. If we consider the case of sensor and *actuator* networks, which may interact with their environment in a closed loop, it will become compulsory to model the environment, as it is the case with any control system, for both functional and non functional aspects.

References

[1] Ilker Demirkol, Fatih Alagz, Hakan Deli, and Cem Ersoy. Wireless sensor networks for intrusion detection: Packet traffic modeling. *IEEE Communications Letters*, 10(1):22–24, January 2006.

[2] Ian Downard. Simulating sensor networks in ns-2, 2005.

[3] Christian C. Enz, Amre El-Hoiydi, Jean-Dominique Decotignie, and Vincent Peiris. Wisenet: An ultralow-power wireless sensor network solution. *IEEE Computer*, 37(8):62–70, 2004.

[4] Victor S. Frost and Benjamin Melamed. Traffic modeling for telecommunications networks. *IEEE Communications Magazine*, pages 70–81, March 1994.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MOBICOM*, pages 56–67, 2000.

[6] Erwan Jahier and Pascal Raymond. The lucky language reference manual. Technical Report TR-2004-6, Verimag Technical Report, 2005.

[7] Sung Park, Andreas Savvides, and Mani B. Srivastava. Sensorsim: a simulation framework for sensor networks. In *MSWIM '00: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 104–111, New York, NY, USA, 2000. ACM Press.

[8] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA, 2004. ACM Press.

[9] Ludovic Samper, Florence Maraninchi, Laurent Mounier, and Louis Mandel. GLONEMO: Global and accurate formal models for the analysis of ad-hoc sensor networks. In *Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06)*, Nice, France, May 2006.

[10] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang. J-sim: A simulation environment for wireless sensor networks. In *Annual Simulation Symposium*, pages 175–187, 2005.

[11] Avinash Sridharan, Marco Zuniga, and Bhaskar Krishnamachari. Integrating environment simulators with network simulators. Technical report, University of Southern California, 2004.

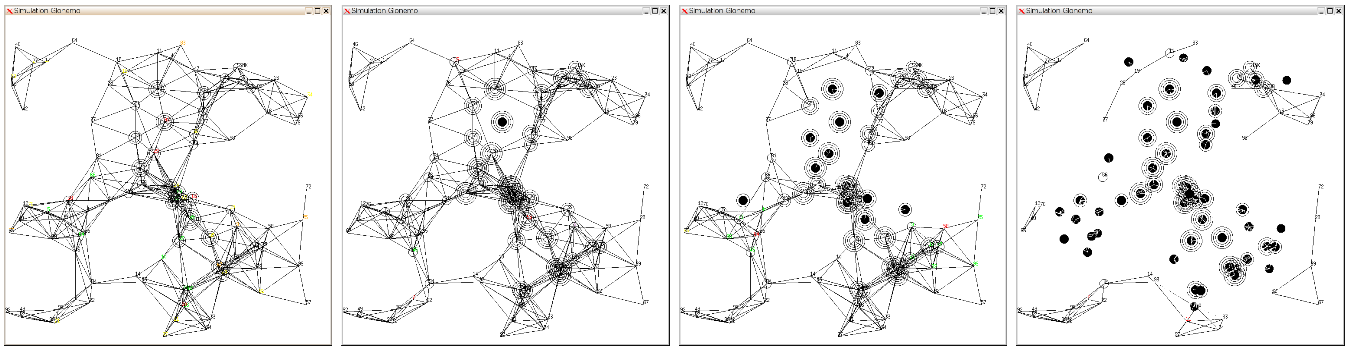


Fig. 6. Picture of a Glonemo simulation with Poisson processes on all nodes.

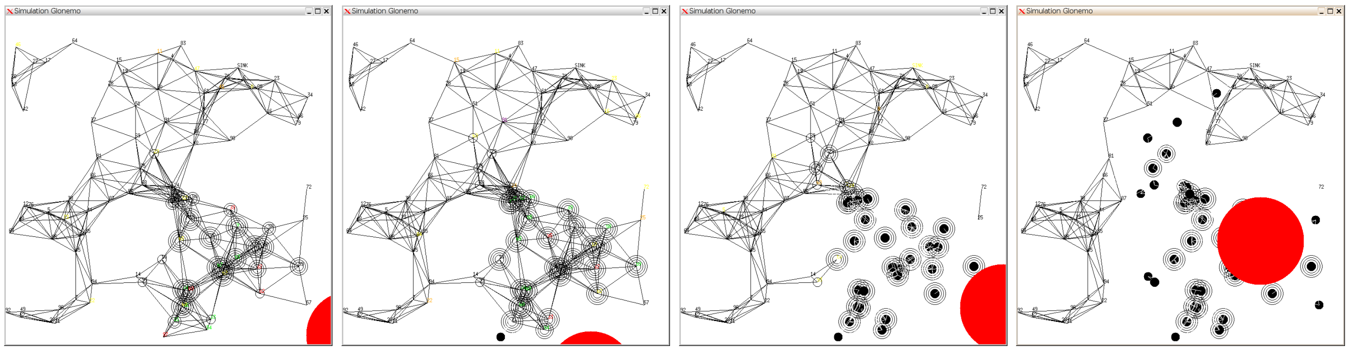


Fig. 7. Picture of a Glonemo simulation with the Lucky cloud. The cloud is confined in the down-right zone.

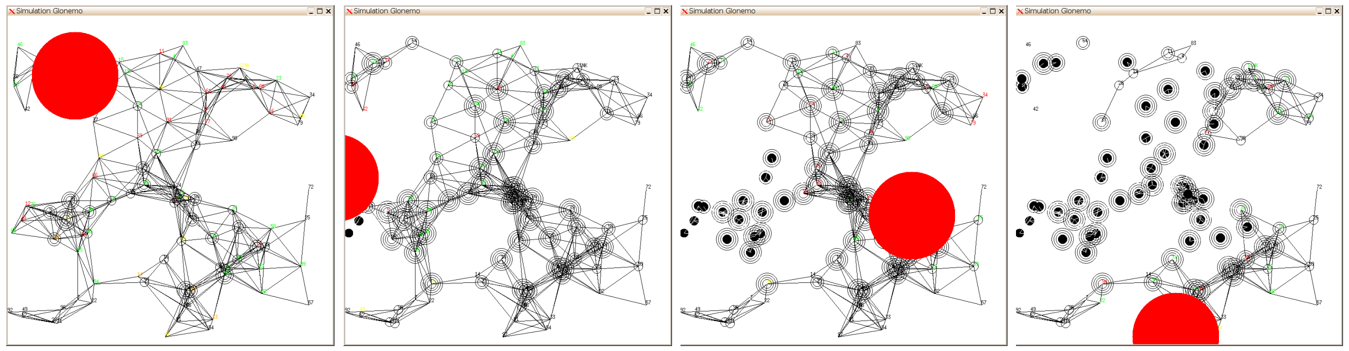


Fig. 8. Picture of a Glonemo simulation with the Lucky cloud. The cloud can be anywhere.

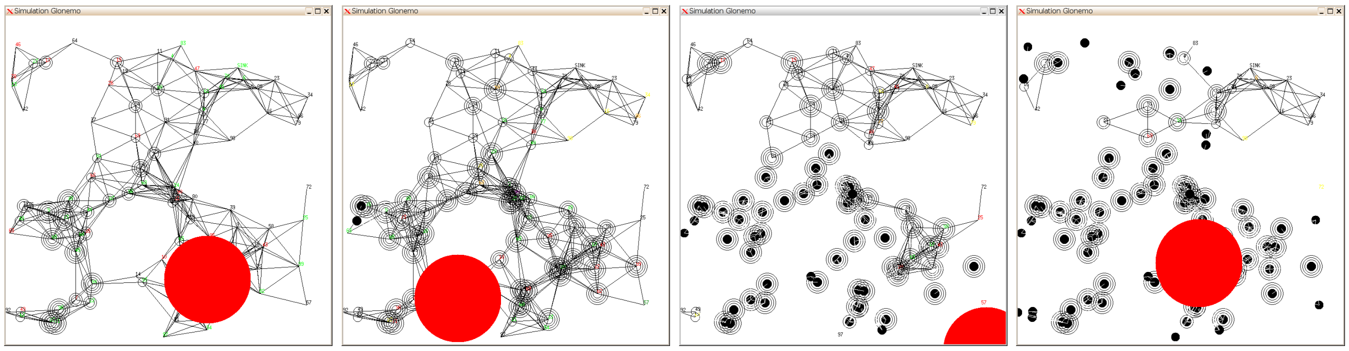


Fig. 9. Picture of a Glonemo simulation with the Lucky cloud. The cloud can be anywhere except on the sink.