Assignment 2 **SYSC5104**

Gutierrez Alcaraz J. Marcelo
Liu Hui

Simulating Wireless Sensor Networks Using Cell-DEVS

Motivation:

Wireless sensor networks (WSN) differ from traditional computer networks in various aspects. Basically, these networks have a great number of distributed nodes (sensor nodes) with the ability to monitor their surroundings. The most important performance aspect in a WSN is the need to be energy efficient as the sensor node has a finite energy reserve offered by a battery.

Conceptual model description:

In this work, a topology control algorithm was implemented for WSNs using Cell-DEVS. The algorithm performs the node scheduling by using the application knowledge and does not address the data communication problem. In the simulator, each cell of the grid corresponds to a sensor node, and each node can have up to eight neighbors. There are three possible states for each node: active, stand-by and dead. In the active state, the node is performing some processing task, and it is monitoring an area equivalent to its neighborhood. In the stand-by mode, the sensor node is saving its energy probably because some other nodes are already monitoring an equivalent area. The dead state means that the sensor has depleted the battery and it’s not useful anymore, the same state applies for cells that don’t have any sensor at all. The main goal is to achieve a longer network lifetime through turning off the nodes that are performing a redundant monitoring task in specific periods of time.

In the proposed algorithm, all the active nodes in the grid must verify, from time to time, if more than one of their neighbors is also active. If there are two or more active neighbors at that moment, the node must change its state to the stand-by mode to save energy. The node will be in the stand-by mode for a random period of time, and after that period, it will “wake up” and make verification. If it has less than two active neighbors at that moment, the node will set its state to active again. After a random time interval, it will perform the neighborhood verification again, and this loop continues until the node runs out of energy.

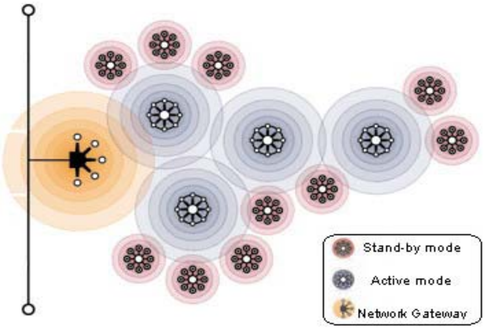


Figure 1. Proposed Network for Simulation

The simulation code as stated in the definition of [1] was implemented in CASim, A pseudo-code of the algorithm can be seen:

Algorithm 1 [1]

Node Scheduling Rule

```
while network is alive do
   for each cell do
      if node is alive (node still has energy) then
         if state is active then
            Decrement node's energy (at active rate)
            Verification
         else
            Decrement node's energy (at stand-by rate)
            Verification
         end if
      end if
   endfor
end while
```

Algorithm 2 [1]

Verification Rule

```
if node's energy is zero then
   Node is dead
else
   if timer is zero then
      Consult CA's rules to retrieve new state
      Re-initiate timer
   else
      Decrement timer
   end if
end if
```

The Model

This work is based in the model simulated in [1] using the CASim Simulator for discrete events, but we will go beyond that work trying to mimic the real expected behaviour of each one of the sensors, because it's easier to test this in a CELL-DEVS simulation platform.

Wireless Sensor Networks are distributed ad-hoc networks that can consist of hundreds or thousands of sensors (nodes) that have the capability of sensing, processing and communicating using a wireless medium [2]. Each of the nodes, besides doing all the common tasks expected from it, has to be energy efficient using full power only in transmission mode, and that happens only when the sensor is part of the network or there are no more sensors around sensing the same data. If there are two or more active sensor neighbours at the same time in the same "environment" (neighbourhood) the sensor should change its state to stand-by mode to save energy and be ready to "come up" when one of the sensors runs out of battery.

These assumptions give as the result the following CELL-DEVS coupled model for each sensor:

WSensor = < I_A, X_A, Y_A, Xlist$_A$, Ylist$_A$, η, N_A, {m_A, n_A}, C_A, B_A, Z_A, select$_A$>
WSensorstate = < I_B, X_A, Y_A, Xlist$_A$, Ylist$_A$, η, N_A, {m_A, n_A}, C_A, B_A, Z_B, select$_B$>

Xlist$_A$ = { 0 };
Ylist$_A$ = { 0 }.
η = 30x30x2;

Z_B:

```
[state-shift-rule]
rule : 0 1000 { (0,0,1) < 1 }
rule : 0 1000 { (0,0,0) = 0 }
rule : 2   {randInt(9)*1000}    { (0,0,0)=1 and (0,0,1)>=1 }
rule : 0   1000                    { (0,0,0)=1 and (0,0,1)<1 }
rule : 2   {randInt(9)*1000}    { (0,0,0)=-1 and (0,0,1)>=1  }
rule : 0   1000                    { (0,0,0)=-1 and (0,0,1)<1 }
rule :  -1  100    { (0,0,0)=2  and statecount(1)>1 }
rule :  1   100    { (0,0,0)=2  and statecount(1)<2 }
```

Z_A:

```
[energy-reduction-rule]
rule : {(0,0,0)*.8 }     1000  { (0,0,0)>1 and (0,0,-1)=1  }
rule : {(0,0,0)*.95 }    1000  { (0,0,0)>1 and (0,0,-1)=-1  }
rule : {(0,0,0) }        100   { (0,0,0)>1 and (0,0,-1)=2  }
rule : 0                 1000  { (0,0,0)>1 and (0,0,-1)=0  }
rule : 0                 1000  { (0,0,0)<=1 }
```

N_A:

```
neighbors : sensor(-1,-1,0) sensor(-1,0,0) sensor(-1,1,0)
neighbors : sensor(0,-1,0)  sensor(0,0,0)  sensor(0,1,0)
neighbors : sensor(1,-1,0)  sensor(1,0,0)  sensor(1,1,0)
neighbors : sensor(0,0,1)
neighbors : sensor(0,0,-1)
```

B_A = {nowrapped};
select$_A$ = { CAij / i Î [1,9], j Î [1,10]} };

S_B:

-1	Stand-by mode
0	Dead or no sensor
1	Active (transmitting)
2	Verification mode

S_A: $x/x \in \{0,10\}$

B = nowrapped

d = inertial delay

τ:		
	random<10	verification and active to stand-by or stand-by to active ()
	100	normal elapsed time
	1000	energy depletion step time

The coupled model represents a matrix of 30 by 30 by 2 cells, where the state change (stand-by, dead, active, verifying) is represented as a second plane, this due to simulation purposes. It represents a 30 by 30 by 2 space that is nonwrapped on the edges, the behaviour near the edges does not represent a case of special consideration. As there is no need to write special rules for the borders none is included.

```
[top]
components : sensor

[sensor]
type : cell
dim : (30,30,2)
delay : inertial
defaultDelayTime : 100
border : nowrapped
neighbors : sensor(-1,-1,0) sensor(-1,0,0) sensor(-1,1,0)
neighbors : sensor(0,-1,0)  sensor(0,0,0)  sensor(0,1,0)
neighbors : sensor(1,-1,0)  sensor(1,0,0)  sensor(1,1,0)
neighbors : sensor(0,0,1)
neighbors : sensor(0,0,-1)
localtransition : state-shift-rule
zone : energy-reduction-rule { (0,0,1)..(29,29,1) }
initialvalue : 0
initialCellsValue : sensor.val

[state-shift-rule]
rule : 0 1000 { (0,0,1) < 1 }
rule : 0 1000 { (0,0,0) = 0 }
rule : 2   {randInt(9)*1000}    { (0,0,0)=1 and (0,0,1)>=1 }
rule : 0   1000                    { (0,0,0)=1 and (0,0,1)<1 }
rule : 2   {randInt(9)*1000}    { (0,0,0)=-1 and (0,0,1)>=1  }
rule : 0   1000                    { (0,0,0)=-1 and (0,0,1)<1 }
rule :  -1  100    { (0,0,0)=2  and statecount(1)>1 }
rule :  1   100    { (0,0,0)=2  and statecount(1)<2 }

[energy-reduction-rule]
rule : {(0,0,0)*.8 }     1000  { (0,0,0)>1 and (0,0,-1)=1  }
rule : {(0,0,0)*.95 }    1000  { (0,0,0)>1 and (0,0,-1)=-1  }
rule : {(0,0,0) }        100   { (0,0,0)>1 and (0,0,-1)=2  }
rule : 0                 1000  { (0,0,0)>1 and (0,0,-1)=0  }
rule : 0                 1000  { (0,0,0)<=1 }
```

An insight to the rules

Cell State plane rules `[state-shift-rule]`

```
rule : 0 1000 { (0,0,1) < 1 }
rule : 0 1000 { (0,0,0) = 0 }
```

This rule forces each cell to be *dead* when it's supposed to be dead

```
rule : 2   {randInt(9)*1000}    { (0,0,0)=1 and (0,0,1)>=1 }
```

Sets the cell in verifying mode

```
rule : 0   1000                    { (0,0,0)=1 and (0,0,1)<1 }
```

forces dead state in a dying cell (running out of resources)

```
rule : 2   {randInt(9)*1000}    { (0,0,0)=-1 and (0,0,1)>=1  }
```

Sets the cell in verifying mode from stand-by mode

```
rule : 0   1000                 { (0,0,0)=-1 and (0,0,1)<1 }
```

Forces dead state in dying cell coming from stand-by mode

```
rule :  -1  100    { (0,0,0)=2  and statecount(1)>1 }
rule :  1   100    { (0,0,0)=2  and statecount(1)<2 }
```

Assigns active or stand-by mode states to each cell.

Cell State plane rules `[energy-reduction-rule]`

```
rule : {(0,0,0)*.8 }     1000  { (0,0,0)>1 and (0,0,-1)=1  }
```

Decreases energy of the active cell by 20%

```
rule : {(0,0,0)*.95 }    1000  { (0,0,0)>1 and (0,0,-1)=-1  }
```

Decreases energy of the stand-by sensor by 5%

```
rule : {(0,0,0) }        100   { (0,0,0)>1 and (0,0,-1)=2  }
```

If the cell is in verifying state it keeps its energy value (shorter full energy time)

```
rule : 0                 1000  { (0,0,0)>1 and (0,0,-1)=0  }
rule : 0                 1000  { (0,0,0)<=1 }
```

Forces sensors to be dead is the energy is in the [0,1] range

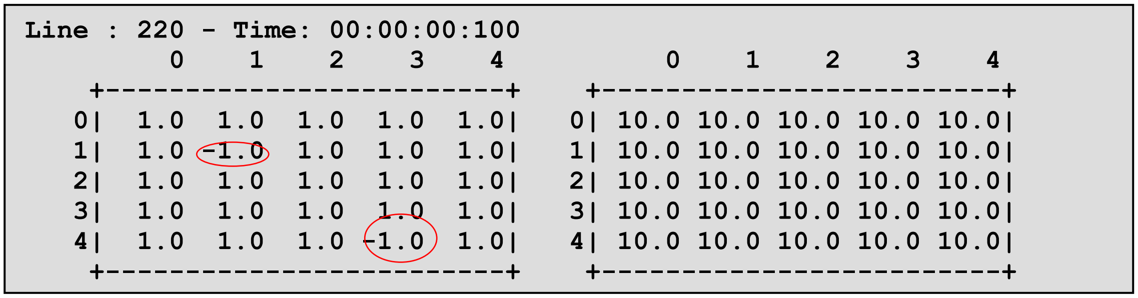
Simulation Results and Testing

The simulation test bed prepared for the network as in [1] deals with more than 2000 sensors simultaneously for the following report we will present the results for a 5x5 network, the results can be extended up to an unlimited number of sensors as can be seen in the .drw files included with this report.

We fed the model with a representation of sensors in every cell but with different (random) states, initially each sensor starts the simulation with full power:

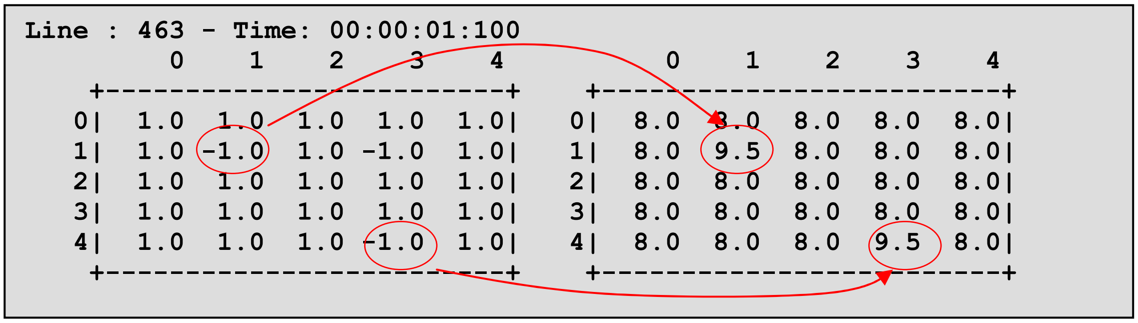
```
Line : 154 - Time: 00:00:00:000
        0    1    2    3    4          0    1    2    3    4
   +-------------------------+    +-------------------------+
  0|  1.0  1.0  1.0  1.0  1.0|   0| 10.0 10.0 10.0 10.0 10.0|
  1|  1.0  2.0  1.0  1.0  1.0|   1| 10.0 10.0 10.0 10.0 10.0|
  2|  1.0  1.0  1.0  1.0  1.0|   2| 10.0 10.0 10.0 10.0 10.0|
  3|  1.0  1.0  1.0  1.0  1.0|   3| 10.0 10.0 10.0 10.0 10.0|
  4|  1.0  1.0  1.0  2.0  1.0|   4| 10.0 10.0 10.0 10.0 10.0|
   +-------------------------+    +-------------------------+
```

The first step of the algorithm will be to change the state of each node after a random time ($\Delta t_{verifying}$) but shorter than the normal time required to deplete 1 unit of the energy resources. Due to the rules stated (2 represents the verifying state) only the nodes that are in verifying state can change their state.



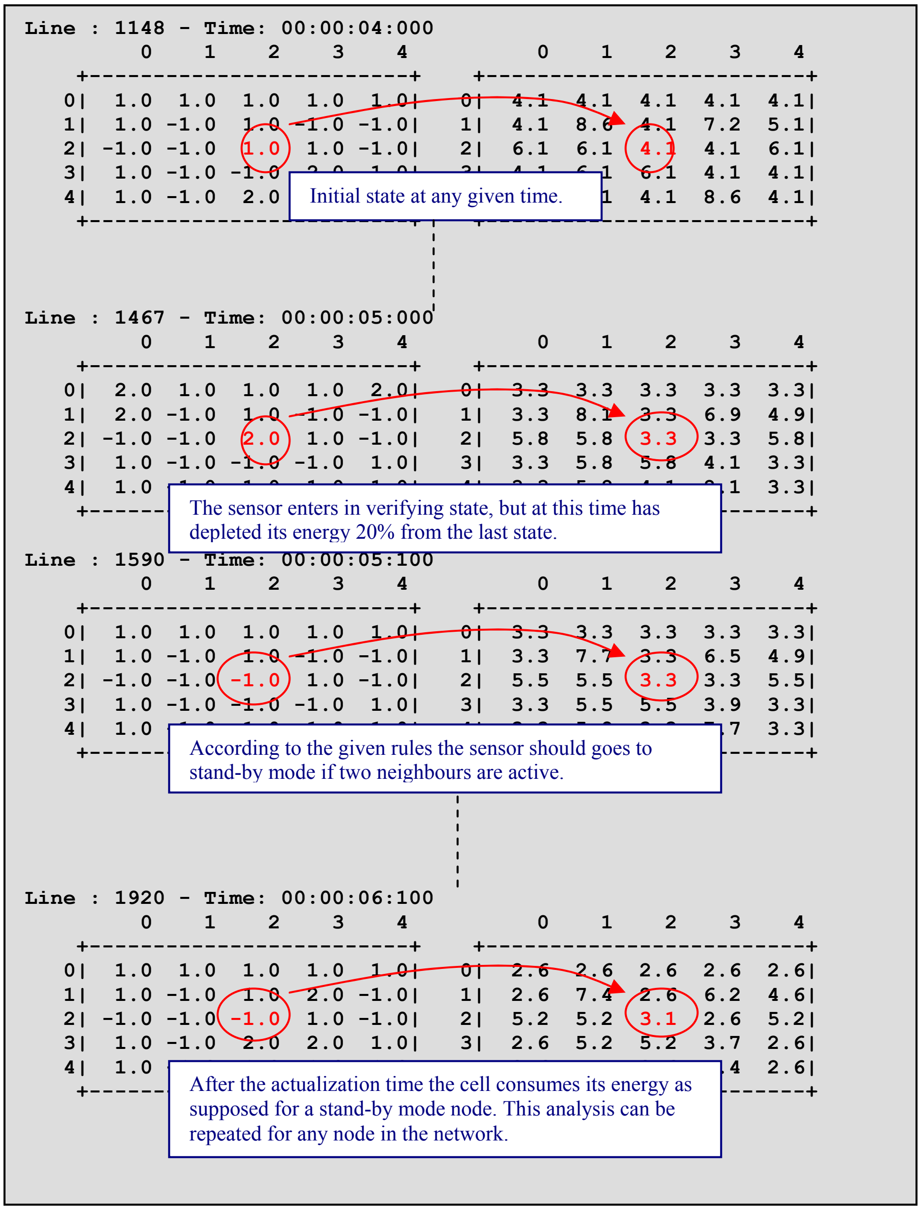
```
Line : 220 - Time: 00:00:00:100
        0    1    2    3    4          0    1    2    3    4
   +-------------------------+    +-------------------------+
  0|  1.0  1.0  1.0  1.0  1.0|   0| 10.0 10.0 10.0 10.0 10.0|
  1|  1.0 -1.0  1.0  1.0  1.0|   1| 10.0 10.0 10.0 10.0 10.0|
  2|  1.0  1.0  1.0  1.0  1.0|   2| 10.0 10.0 10.0 10.0 10.0|
  3|  1.0  1.0  1.0  1.0  1.0|   3| 10.0 10.0 10.0 10.0 10.0|
  4|  1.0  1.0  1.0 -1.0  1.0|   4| 10.0 10.0 10.0 10.0 10.0|
   +-------------------------+    +-------------------------+
```

The next simulation step calculates the energy consumption based on the state of each sensor, an active state sensor should deplete their energy resources at a rate of 20% and a stand-be mode sensor should do it at 5% (there is a verifying result snapshot before this snapshot that is not shown):



```
Line : 463 - Time: 00:00:01:100
        0    1    2    3    4          0    1    2    3    4
   +-------------------------+    +-------------------------+
  0|  1.0  1.0  1.0  1.0  1.0|   0|  8.0  8.0  8.0  8.0  8.0|
  1|  1.0 -1.0  1.0 -1.0  1.0|   1|  8.0  9.5  8.0  8.0  8.0|
  2|  1.0  1.0  1.0  1.0  1.0|   2|  8.0  8.0  8.0  8.0  8.0|
  3|  1.0  1.0  1.0  1.0  1.0|   3|  8.0  8.0  8.0  8.0  8.0|
  4|  1.0  1.0  1.0 -1.0  1.0|   4|  8.0  8.0  8.0  9.5  8.0|
   +-------------------------+    +-------------------------+
```

Due to the dynamic nature of the network is not possible to track all of the nodes accurately (each nodes goes to and comes from stand-by mode depending on the state of the surrounding nodes), but it’s possible to verify the algorithm and the simulation in just one particular node, for testing purposes we chose the sensor located in the (2,2) cell.



```
Line : 1148 - Time: 00:00:04:000
        0    1    2    3    4          0    1    2    3    4
    +-------------------------+    +-------------------------+
   0|  1.0  1.0  1.0  1.0  1.0|   0|  4.1  4.1  4.1  4.1  4.1|
   1|  1.0 -1.0  1.0 -1.0 -1.0|   1|  4.1  8.6  4.1  7.2  5.1|
   2| -1.0 -1.0  1.0  1.0 -1.0|   2|  6.1  6.1  4.1  4.1  6.1|
   3|  1.0 -1.0 -1.0                         6.1  4.1  4.1|
   4|  1.0 -1.0  2.0                    1  4.1  8.6  4.1|
    +----------------                   -----------------+
```

Initial state at any given time.

```
Line : 1467 - Time: 00:00:05:000
        0    1    2    3    4          0    1    2    3    4
    +-------------------------+    +-------------------------+
   0|  2.0  1.0  1.0  1.0  2.0|   0|  3.3  3.3  3.3  3.3  3.3|
   1|  2.0 -1.0  1.0 -1.0 -1.0|   1|  3.3  8.1  3.3  6.9  4.9|
   2| -1.0 -1.0  2.0  1.0 -1.0|   2|  5.8  5.8  3.3  3.3  5.8|
   3|  1.0 -1.0 -1.0 -1.0  1.0|   3|  3.3  5.8  5.8  4.1  3.3|
   4|  1.0                                         .1  3.3|
    +------                                         ------+
```

The sensor enters in verifying state, but at this time has depleted its energy 20% from the last state.

```
Line : 1590 - Time: 00:00:05:100
        0    1    2    3    4          0    1    2    3    4
    +-------------------------+    +-------------------------+
   0|  1.0  1.0  1.0  1.0  1.0|   0|  3.3  3.3  3.3  3.3  3.3|
   1|  1.0 -1.0  1.0 -1.0 -1.0|   1|  3.3  7.7  3.3  6.5  4.9|
   2| -1.0 -1.0 -1.0  1.0 -1.0|   2|  5.5  5.5  3.3  3.3  5.5|
   3|  1.0 -1.0 -1.0 -1.0  1.0|   3|  3.3  5.5  5.5  3.9  3.3|
   4|  1.0                                         .7  3.3|
    +------                                         ------+
```

According to the given rules the sensor should goes to stand-by mode if two neighbours are active.

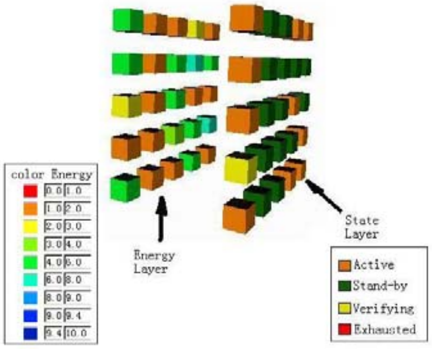
```
Line : 1920 - Time: 00:00:06:100
        0    1    2    3    4          0    1    2    3    4
    +-------------------------+    +-------------------------+
   0|  1.0  1.0  1.0  1.0  1.0|   0|  2.6  2.6  2.6  2.6  2.6|
   1|  1.0 -1.0  1.0  2.0 -1.0|   1|  2.6  7.4  2.6  6.2  4.6|
   2| -1.0 -1.0 -1.0  1.0 -1.0|   2|  5.2  5.2  3.1  2.6  5.2|
   3|  1.0 -1.0  2.0  2.0  1.0|   3|  2.6  5.2  5.2  3.7  2.6|
   4|  1.0                                         .4  2.6|
    +------                                         ------+
```

After the actualization time the cell consumes its energy as supposed for a stand-by mode node. This analysis can be repeated for any node in the network.

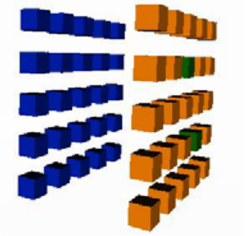
WSN – VRML Graphics

There were some problems with the exportation of CD++ data to the VRML online application due to some java plug-in problems, this was solved with the given java application.

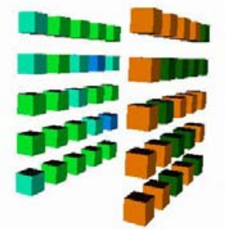
The following pictures are the results obtained by using the vrml graphics generation of the simulated WSN model.

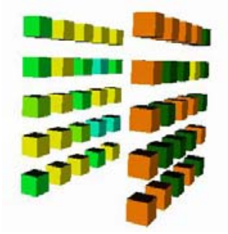


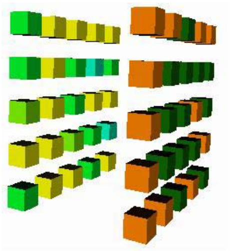
One layer is used to set the states of the cell and the other one represents cell energy.

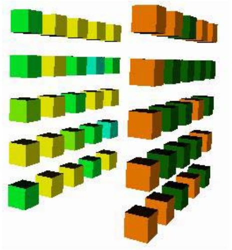


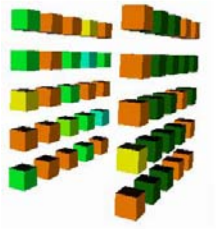
Initial snapshot

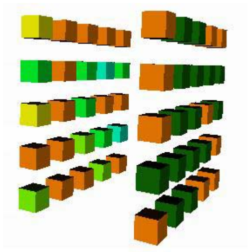


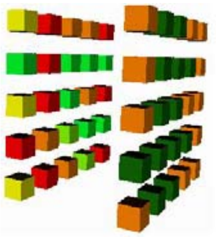












Red represents a depleted cell

Conclusion

With the present work we obtained the same results as in [1] although we didn't run the same amount of simultaneous sensors (up to 2500) due to time constraints. But we have checked the algorithm in a step by step basis and it is easy to infer from it that the same results will be generated.

With the additional benefit that it is easier to simulate and test any algorithm in a CELL-DEVS tool and obtain the same results as in some other simulators with some more implementation complexity.

References

1 **Simulating Large Wireless Sensor Networks Using Cellular Automata**

Renan Cunha, Aloizio Silva, Antonio F. Loureiro, Linnyer B. Ruiz
IEEE Proceedings of the 38th Annual Simulation Symposium (ANSS'05)
2005

Attached as appendix A
2 **Cell-DEVS Class Notes**

Gabriel Winer, Fall 2005, Carleton University.
3 **CD++ User's Guide**

Daniel A. Rodriguez, Gabriel A. Wainer
Departamento de Computación, Universidad de Buenos Aires
Buenos Aires, Argentina
1999