Emergent Behavior in Massively-Deployed Sensor Networks

Ekaterina Shurkova 1), Ruzana Ishak 2), Stephan Olariu 1), Shaharuddin Salleh 3)

Abstract
The phenomenal advances in MEMS and nanotechnology make it feasible to build small devices, referred to as sensors that are able to sense, compute and communicate over small distances. The massive deployment of these small devices raises the fascinating question of whether or not the sensors, as a collectivity, will display emergent behavior, just as living organisms do. In this work we report on a recent effort intended to observe emerging behavior of large groups of sensor nodes, like living cells demonstrate.

Imagine a massive deployment of sensors that can be in two states “red” and “blue”. At deployment time individual sensors have an initial color. The goal is to obtain a uniform coloring of the deployment area. Importantly, the sensors can only talk to sensors that are one-hop away from them. The decisions to change colors are local, based on what the sensors can infer from collecting color information from their neighbors. We have performed extensive simulations involving 20,000 sensors in an area of 100m × 100m. Our simulation results show that the sensor network converges to a stable uniform coloring extremely fast.

1. Introduction

Through the history of humankind people always try to copy nature’s inventions and to borrow nature’s developments. From animal furs for keeping warm people came to aircrafts for flying. Everything that human beings know and discover exists in nature. Now humans try to make one big step further – to simulate the biggest secret of nature, to emulate life itself. In biology it is cloning, in exact sciences it is artificial intelligence, intelligent robots and now a new project inspired by nature – observing behavior of massively deployed sensor networks, with the goal to compare their behavior with the behavior of living cell colonies.

Such devices (sensor nodes) are able to sense small area parameters (e.g. temperature, humidity, etc) and communicate with their closest neighbors in this type of community using short-range radio. Based on sensed and gathered from this community information they can make some assessments concerning surrounding environment, relations between neighbors, etc. Since in the last two decades the main interest of researchers in the sensor network field was center-controlled

1 Department of Computer Science, Old Dominion University, Norfolk, Virginia 33529, USA, eshrkova@cs.odu.edu, olariu@cs.odu.edu
2 Department of Science, College of Science and Technology, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia, ruzana@citycampus.utm.my, ruzanaishak@yahoo.com
3 Department of Mathematics, Faculty of science, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia, ss@mel.fs.utm.my
groups of sensors, it is important for this work to emphasize the concept of *collectiveness*. Centralized control assumes existence of an entity that is in some sense different from all other sensors and which has some kind of “power” over the other nodes, generally speaking it plays an exclusive role. It either has more information (accumulated from other sensors or pre-given) or it has an ability to influence the rest of the sensors in their activities: train them, put them to sleep, awake them for some activity, etc. However these days the new point of interest in sensor networks is the research work which has a purpose of discovering a collective behavior. As opposed to the centralized control, living cells demonstrate collective or *emerging* behavior. There is no “big brother” cell which gives orders. It is a genetic material embedded within a cell so that it knows which activity to perform, how to communicate with the neighbors, how to interpret their responses. The interesting fact is that the emerging behavior is a product of cells intercommunication, of a collective inter-performance, and it cannot be seen in a single living cell.

2. The Contribution

Our contribution of this work is to deploy “red” and “blue” sensor nodes and to let them collectively distribute themselves to get “blue” and “red” color sensors uniformly spread out around the field by assuming that sensors are not synchronized in time. For example, the “wolves-sheep” problem. An environment of these two species is stable only when the ratio of numbers of these species is within a certain range. The issues appear when:

- *there are too many wolves* (at some point they will eat the entire population of sheep which will cause extinction of wolves or a need to change a location of wolves pack);
- *there are too many sheep* (at some point in time they will eat out all the grass in the habitat area which will cause their extinction or a need for changing location);

By taking into account of the “population” of technical devices, we imagine a lawn and a “colony” of smart sprinklers which are capable to sense local area, to analyze humidity of the area and to make a decision either to: turn myself on or turn myself off.

It is obvious that in such a model it is ideal for the lawn to have a stability of turned on and turned off sprinklers at some point of time in some particular region. Theoretically having stability locally this model should also have stability at the global level.

![Figure 1. Smart sprinklers population](image-url)

3.1. Sensor Node

Sensor node (sensor) is a device which has a sensor to measure certain environment characteristics (e.g. temperature, humidity, etc), a processor which performs calculations and a short-range transceiver device which allows sensor to communicate with its immediate neighbors. Each sensor has genetic material (information given to sensor node at manufacturing). In this work genetic material is considered to be:

- color – red or blue (sensors are conditionally divided into two groups: each sensor is either “red” or blue; application of this idea is for example to divide sensors by duties);
- status – phase of life cycle period;
- timing of each life cycle phase;
- local clock;
- ability to calculate neighborhood coloring scheme (e.g. 2 reds, 5 blues);
- ability to change color based on neighborhood coloring scheme (e.g. if I’m “blue” and I hear 2 reds and 5 blues, I need to change a color);
- transmission range – radius of radio transmission;
- conservativity

3.2. Sensor Node Life Cycle

Sensor’s life time consists of four phases:
- phase of sleeping (s);
- listen-to-reds-phase (LR);
- listen-to-blues phase (LB);
- transmitting phase (t).

They occur in this exact sequence and compose one life cycle. Life cycles come one after another. As it is implied from the names of phases:
- during phase of sleeping sensor node does nothing; it helps sensor to save power;
- during listen-to-reds phase sensor tunes its radio-receiver to the frequency of “red” sensors
  - a particular range of transmitting frequency which can be used by only red-colored sensors;
  - sensor gathers information from its “red” neighbors during this phase;
- during listen-to-blues phase sensor tunes its radio-receiver to the frequency of “blue” sensors
  - a particular range of transmitting frequency which can be used by only blue-colored sensors;
  - sensor gathers information from its “blue” neighbors during this phase;
- during transmitting phase sensor tunes its radio-transmitter to the frequency of its color (e.g. if node is “red” chosen frequency should be one which is reserved for “red” sensors only) and transmits a message to its neighbors; a message will be defined later on.
3.3. Sensor Setup

At the deployment moment sensors need to initialize themselves. Sensors configuration was described earlier (paragraph 3.1). Each sensor node has to start its life cycle and set up all its characteristics to specific values. Since in this work synchronization of sensor nodes is not considered, each sensor starts to sleep, listen and transmit at an individual time and has an individual period of time for each activity.

Sensor’s life cycle (see 3.1) consists of four alternating stages: sleep, listen-to-reds, listen-to-blues, and transmit. At the “birth” moment sensor does what it is programmed to do – it chooses the values for the following parameters:

- “time-to-sleep” – for how many time units this sensor node will be sleeping;
- “time-to-listen-to-reds” - for how many time units sensor’s receiver will be tuned to the “red” frequency;
- “time-to-listen-to-blues” - for how many time units sensor’s receiver will be tuned to the “blue” frequency;
- “time-to-transmit” - for how many time units sensor’s transmitter will be tuned to a particular color frequency to transmit message to its neighborhood.

In this work it was suggested that the following values will lead to the intuitively predicted realistic results (sensor node needs to choose exactly one value of those listed in brackets):

- “time-to-sleep”: {60, 61, 62};
- “time-to-listen-to-reds”: {3, 4, 5};
- “time-to-listen-to-blues”: {3, 4, 5};
- “time-to-transmit”: {3, 4, 5};

For example, at the deployment moment one of the sensors chooses the following scheme: (61, 4, 3, 5) which means this sensor will sleep for 61 time units, then it will be in a listening-to-reds mode for 4 units of time, then it will come to listening-to-blues stage for 3 time units, and finally to a transmitting mode for 5 time units. This life cycle is illustrated by Figure 3.

```
1 2 3 ... 61 1 2 3 4 1 2 3 1 2 3 4 5
Sleep Listen-to-reds Listen-to-blues Transmit
```

Figure 3. One life cycle of a sensor node

Now this sensor has a defined life cycle and it needs to choose the “entry point”. It randomly seats itself in a life cycle timing interval. It “flips a coin” – randomly generates a number between 1 and (61 + 4 + 3 + 5) = 73. Assume it chose 19. It means its STATUS at the beginning is SLEEP and its local clock is 19. Figure 3 shows an example of the initialization of four different randomly chosen sensor nodes.
In this work, Transmission Range was chosen to be 1 unit, which means sensor's message over the radio can reach its neighbors within a circle of radius 1. Transmission range here is actually an interaction range since this is a circle within which sensors in an immediate neighborhood can “hear” each other.

3.4. Sensors Interaction

As illustrated by Figure 6 during a sleeping period sensor node does nothing – it is a phase for saving power. During next – listen-to-reds phase – sensor accumulates information about its “red” neighbors. It listens to its reachable within transmission range neighbors and counts how many of them it can hear during this period. The same information (but about “blue” immediate neighbors) sensor gets during the listen-to-blues stage of its life cycle. After that sensor has all information about its neighborhood and it has to make a comparison – how balanced is the coloring scheme of its surrounding area? After that it needs to make a decision on its own color. For example, if it is “blue” and it has too many “blue” neighbors, it should change its color to red. Right after that sensor starts transmitting on a particular frequency a message which contains sensor’s color.

3.5. Conservativity

Once sensor has accumulated the information about “red” and “blue” neighbors, it calculates the balance of the area coloring scheme within its transmission range:
The balance is the number between -1 and 1, since:

- if blues# = 0 (no “blue” neighbors), then balance = -1;
- if reds# = 0 (no “red” neighbors), then balance = 1;
- if balance = 0, then the neighborhood is uniformly colored (ideal situation).

Since sensor gets information from the awake AND transmitting sensors, this information is not accurate enough – some of the neighbors are listening and some of them are sleeping. To handle such inaccuracy a sensor must be “conservative”:

- if balance is in range [-0.25; 0.25] sensor assumes that the neighborhood is stable enough and does not change the color;
- if balance > 0.25, it means that there are too many “blue” sensors within its transmission range and if the sensor is blue, it should change its color;
- if balance < -0.25, it means that there are too many “red” sensors within its transmission range and if the sensor is red, it should change its color;
- otherwise the decision is not to change color.

Conservativity is illustrated by Figure 6.

Figure 6. Conservativity

4. Implementation and Simulation Results

4.1. Implementation

Red-Blue Wireless Network Simulation is implemented in Visual C++ using Microsoft Visual Studio 6.0 environment. Implementation consists of two main classes: class sensor and class environment and structure struct message. Class sensor is implemented in such a way that sensor itself is not aware neither of its location on the field nor of the network structure. Sensor has genetic material (as described in section 3.1) consisting of its color; list of phases sensor goes through during the simulation; sensor’s own clock which is being initialized to zero every time when current phase (s, LR, LB, t) is changed to the next one; function which computes neighborhood coloring scheme each time when sensor is in a transit from LB stage to t stage; function which “makes decision” whether to change or not sensor’s color. Class sensor has access to struct message. In the beginning of the t phase sensor constructs a new message which is forwarded to the class environment and then environment takes care of it – forwards this message to other sensors in the
transmitting sensor neighborhood which are in the listening mode \((LR\) or \(LB\) depending on the transmitting sensor color).

**Class environment** is a model of the sensors world. It is aware of all the sensors coordinates to responsible of global (for each particular simulation) clock, and handling communication between the sensors. When a sensor transmit a message out into the “air”, it does care how, where and to which sensor it will be delivered – that is the job of the environment. For visualizing sensors population a graphical interface was developed. Figure 6 is a picture of software interface window which is designed for data input and display of the output.

### 4.2. Simulation results

The very first simulation was done on randomly deployed sensors. They were placed at randomly picked coordinates through the field \((10\times10\) units) and sensors were programmed in such a way that they chose their colors randomly. Simulation ran on 20,000 sensors. Figures 8a and 8b show the result of such simulation. Plotted value is a *balance* between “red” and “blue” sensors from the global point of view:

\[
balance = \frac{\text{blues#} - \text{reds#}}{\text{blues#} + \text{reds#}}
\]

![Figure 7. Graphical Interface](image)

![Figure 8a. Randomly deployed sensors](image)
Figures 8a and 8b show the result of such simulation. Plotted value is a *balance* between “red” and “blue” sensors from the global point of view:

\[
balance = \frac{\text{blues} - \text{reds}}{\text{blues} + \text{reds}}
\]

Since there are only two colors to choose from and 20,000 entities, then by the Law of Large Numbers which says:

- *the average of a random sample from a large population is likely to be close to the mean of the whole population;*

the population will have roughly the same number of “red” and “blue” sensors distributed through the field. It is obvious that sensors are already uniformly distributed and the goal is reached.

The next simulation to run was one involving a particular number of “red” and “blue” sensors randomly (on coordinates) distributed on the field. Figure 9a shows the result of simulation for 15,000 “red” sensors and 5,000 “blue” sensors. Figure 9b shows the “opposite” case: 5,000 “red” sensor nodes and 15,000 “blue” ones.
Since these two cases are similar from the point of view of the network (global) color balance, one can take a closer look at one of these situations and the second one can be explained in the same manner. Therefore, without sacrificing the generality, this paragraph will investigate the deployment of 15,000 “reds” and 5,000 “blues”. At the deployment moment there was a serious numeral superiority of “red” sensors (Figure 9a). Why does the plot continue to grow to the side of “red” sensors?

Deployment statistics gives an explanation. As it was described before each sensor has a life cycle of sleeping phase (61 to 63 units), listen-to-reds phase (3 to 5 time units), listen-to-blues phase (3 to 5 time units), and transmitting phase (3 to 5 time units), therefore each sensor node has a life cycle of 70 – 78 time units. At the “birth” moment sensor picks a “set up point” – when to start – and it sits itself randomly within a life cycle timing interval. The Law of Large Numbers mentioned earlier, works at this point also: the majority of sensors will pick “entry point” at the sleeping period since this period is the longest one. Table 1 shows sensors statistics at the deployment moment.

### Table 1. Statistics at the deployment moment. Five different experiments

<table>
<thead>
<tr>
<th>Exp #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>sleep</td>
<td>16,768</td>
<td>16,752</td>
<td>16,808</td>
<td>16,716</td>
<td>16,735</td>
</tr>
<tr>
<td>L/R</td>
<td>1,097</td>
<td>1,100</td>
<td>1,070</td>
<td>1,111</td>
<td>1,114</td>
</tr>
<tr>
<td>L/B</td>
<td>1,096</td>
<td>1,081</td>
<td>1,053</td>
<td>1,101</td>
<td>1,084</td>
</tr>
<tr>
<td>transmit</td>
<td>1,079</td>
<td>1,057</td>
<td>1,069</td>
<td>1,072</td>
<td>1,061</td>
</tr>
</tbody>
</table>

Furthermore, on a field of 10x10 units there are 20,000 sensors deployed; transmission range is 1 unit. As one can see from Table 1, on average there are approximately 16,700 of sensors sleeping at this time, which implies that approximately 3,300 nodes are awake, which means they are either listening or transmitting. Since all sensors were deployed randomly, it is assumed that these 3,300 awake sensors are also distributed randomly and roughly uniformly. Therefore on a field of 10x10 units there are about 3,300 awake sensors, meaning approximately 33 awake sensors per 1 unit². The area of communication region for each sensor is: \( \pi \times TR^2 \approx 3.14 \times 1 = 3.14 \text{ units}^2 \). Thus, \( 3.14 \times 33 \approx 104 \) sensors are within a transmission range (provided the deployment distribution is uniform) of one sensor. These 104 nodes are awake, meaning that they belong to four different groups: 1)
listening to reds, 2) listening to blues, 3) transmitting “red” sensors, and 4) transmitting “blue” sensors.

It is not so critical to determine exactly how many transmitting sensors of each color there are, but it is obvious that there are too many of them to get a clear transmission! Therefore, there will be too many collisions. Why is that a problem? Initially there were 15,000 “red” nodes and 5,000 “blue” sensor nodes. From Table 2 one can see the relation between sensors at different stages:

<table>
<thead>
<tr>
<th></th>
<th>total</th>
<th>Reda</th>
<th>Blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>16,768</td>
<td>12,576</td>
<td>4,192</td>
</tr>
<tr>
<td>L/R</td>
<td>1,057</td>
<td>792</td>
<td>265</td>
</tr>
<tr>
<td>L/B</td>
<td>1,096</td>
<td>822</td>
<td>274</td>
</tr>
<tr>
<td>Transmit</td>
<td>1,079</td>
<td>809</td>
<td>270</td>
</tr>
</tbody>
</table>

Table 2. Red-Blue ratio at the deployment moment

Using this table it can be derived:

- 270 transmitting “blue” sensors per 100 units² and each sensor has communication range (transmission circle) of 3.14 units² ⇒ approximately 8 transmitting “blue” sensors per each sensor’s communication range;
- 809 transmitting “red” sensors per 100 units² and each sensor have communication range of 3.14 units² ⇒ approximately 25 transmitting “red” sensors;

8 is a quite small number and since sensors are distributed randomly it is possible that at some point (very rarely!) the number of transmitting “blue” sensors is reduced to exactly one within this transmission range – in such case this transmitting sensor can be heard by listening ones! In other cases, when two or more sensors transmit, listening sensors can hear only noise because of collisions. Average number of “red” transmitting sensors in the neighborhood is 25 and it is doubtful that at some point there will be exactly one transmitting sensor to be heard. In other words, an event of a “blue” transmitting sensor to be heard is very rare, but an event of a “red” transmitting sensor to be heard is practically impossible. Therefore, listening sensors can hear “blue” sensors sometimes, but they do not hear “red” sensors at all – because of collisions! Thus, they “calculate” that there are too many “blue” sensors in the neighborhood and no “red” neighbors at all – they turn to red color collectively.

5. Conclusions

In this paper, a network of massively deployed sensors was considered as a population of small entities and the main goal of this project was to discover similarities between population of sensor nodes and colonies of cells. To reach this goal software was designed and implemented, a bunch of simulations were conducted to study the influence of different sensor nodes parameters on the overall performance of the population.

Several interesting features of the sensors behavior were discovered that allowed the improvement of the overall emerging behavior of the sensors population. It was detected that in order to be more contributing to the communal longing for the balance, a sensor should be tolerant (conservative)
enough. Its conservativity should be chosen from the “golden ratio” point of view: not too much and not too little. Another aspect discovered while studying the simulation results was that the interference in communication between the sensors can carry misleading information, but with a clever approach of reducing it, collisions can be actually used for improving the accuracy of computations. Thus, instead of “throwing away” the noise sensor gets instead of a clear message, one can make a conclusion from it and use it for good.

6. Future Work

This project was considered as a first step toward solving a real world problem of clock synchronization in wireless sensor networks. It would be quite interesting to see if using the approach and methods of this project one can achieve local synchronization in WSN and even more challenging – if using such locally synchronized neighborhoods of sensors one can solve WSN global synchronization issue.

References


