Mobile Information Systems 4 (2008) 313–331
IOS Press

Emergent behavior in massively-deployed sensor networks

Ekaterina Shurkova[a], Ruzana Ishak[b,*], Stephan Olariu[a] and Shaharuddin Salleh[c]
[a]*Department of Computer Science, Old Dominion University, Norfolk, VA, USA*
[b]*Department of Science, College of Science and Technology, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia*
[c]*Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia, Johor Bahru, Malaysia*

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 100 m $\times$ 100 m. Our simulation results show that the sensor network converges to a stable uniform coloring extremely fast.

Keywords: MEMS, sensors, massively deployed, emerging behaviour, uniform coloring

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) which are produced from the MEMS technology 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 [1,5,6]. Based on sensed and gathered from this community information they can make some assessments concerning surrounding environment, relations between

*Corresponding author: Ruzana Ishak, Department of Science, College of Science and Technology, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia. Tel.: +603 012 298 2650; Fax: +603 2693 4844; E-mail: ruzana@citycampus.utm.my.

neighbors, etc. Since in the last two decades the main interest of researchers in the sensor network field was center-controlled 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 [2,4,8,12]. 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 others 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 "wolfs-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 wolfs (at some point they will eat the entire population of sheep which will cause extinction of wolfs or a need to change a location of wolfs 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, refer to Fig. 1, 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.

3. Red-blue wireless sensor network

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);

Fig. 1. Smart sprinklers population.

- 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;

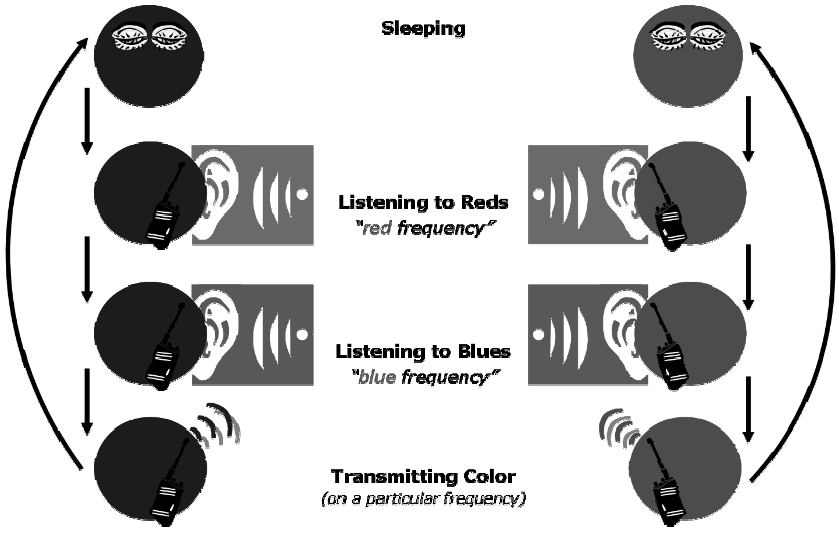


Fig. 2. Sensor life cycle.

– 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 network model

At the deployment moment sensors need to be initialized. Sensors configuration was described earlier in Section 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 in Fig. 2 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 model, 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": $\{3, 4, 5\}$;

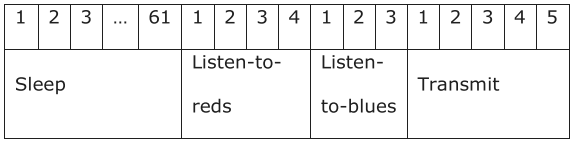


Fig. 3. One life cycle of a sensor node.

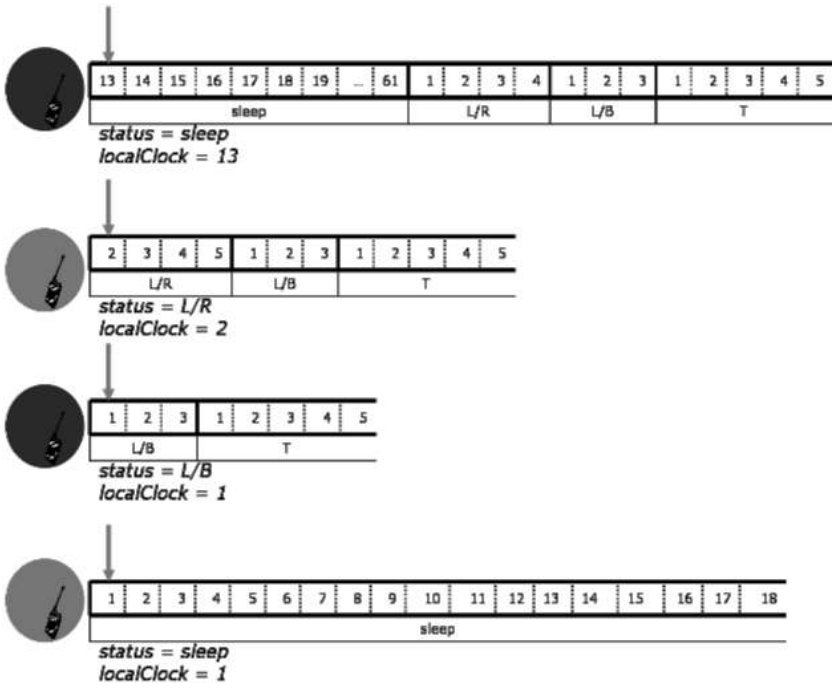


Fig. 4. Sensors set up themselves at the deployment moment.

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 Fig. 3.

After this sensor has a defined life cycle, 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 4 shows an example of the initialization of four different randomly chosen sensor nodes.

In this model, *Transmission Range* was chosen to be 1 unit as shown in Fig. 5, which means sensor's message over the radio can reach its neighbors within a circle of radius 1. Transmission range is an interaction range in a circle within which sensors in an immediate neighborhood can "hear" each other.

3.4. Sensors interaction

As illustrated by Fig. 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

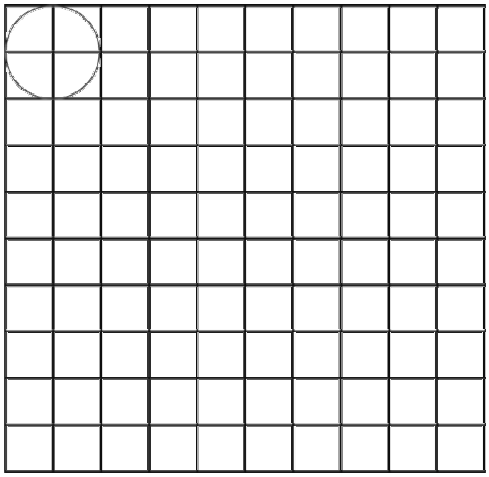


Fig. 5. Communication (Transmission) range of the sensor compared to the entire deployment field.

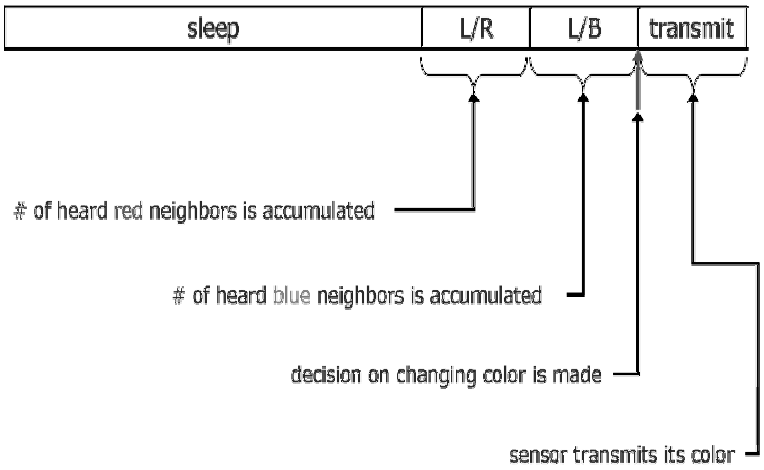


Fig. 6. Life cycle details.

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:

$$balance = \frac{blues\# \ - \ reds\#}{blues\# \ + \ reds\#}$$

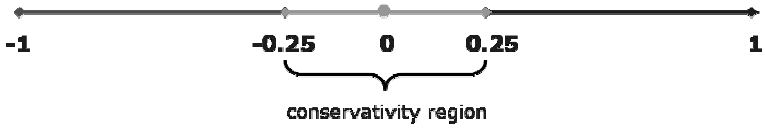


Fig. 7. Conservativity.

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 Fig. 7.

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 sensor's color or not to change. *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 8 is a picture of software interface window which is designed for data input and display of the output.

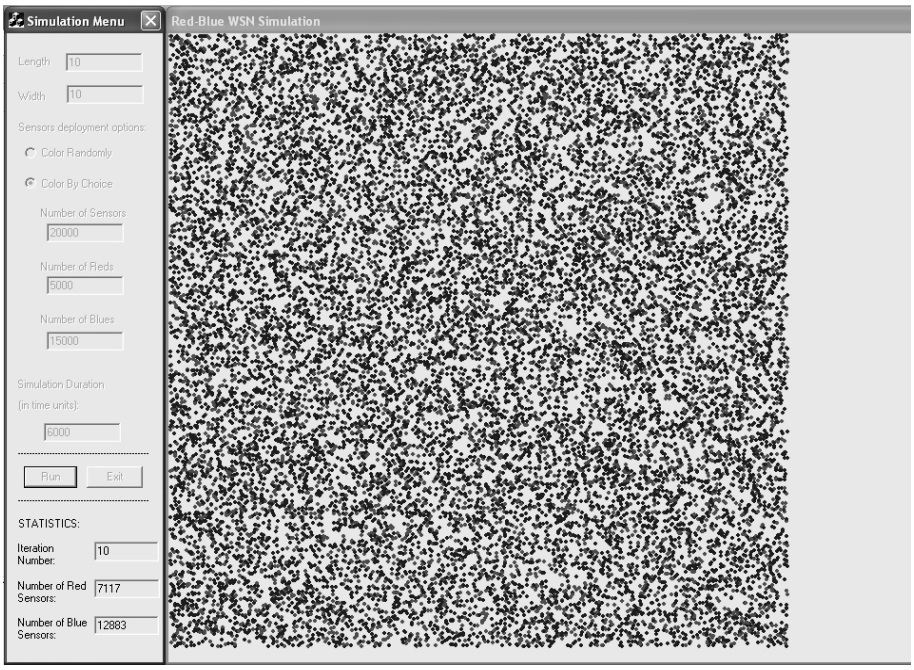


Fig. 8. Graphical interface.

4.2. Simulation results

The very first simulation was done on randomly deployed sensors. They were placed at randomly picked coordinates through the field (10×10 units) and sensors were programmed in such a way that they chose their colors randomly. Simulation ran on 20,000 sensors. Figure 9a and 9b show the result of such simulation. Plotted value is a *balance* between "red" and "blue" sensors from the global point of view:

$$balance = \frac{blues\# \ - \ reds\#}{blues\# \ + \ reds\#}$$

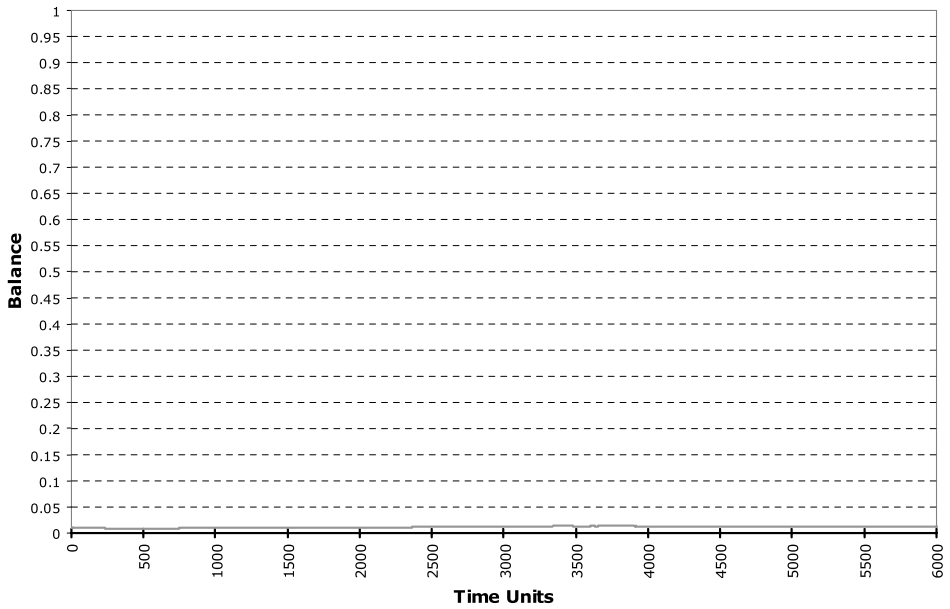
Since there are only two colors to choose from and 20,000 entities, then by the Law of Large Numbers which says:

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

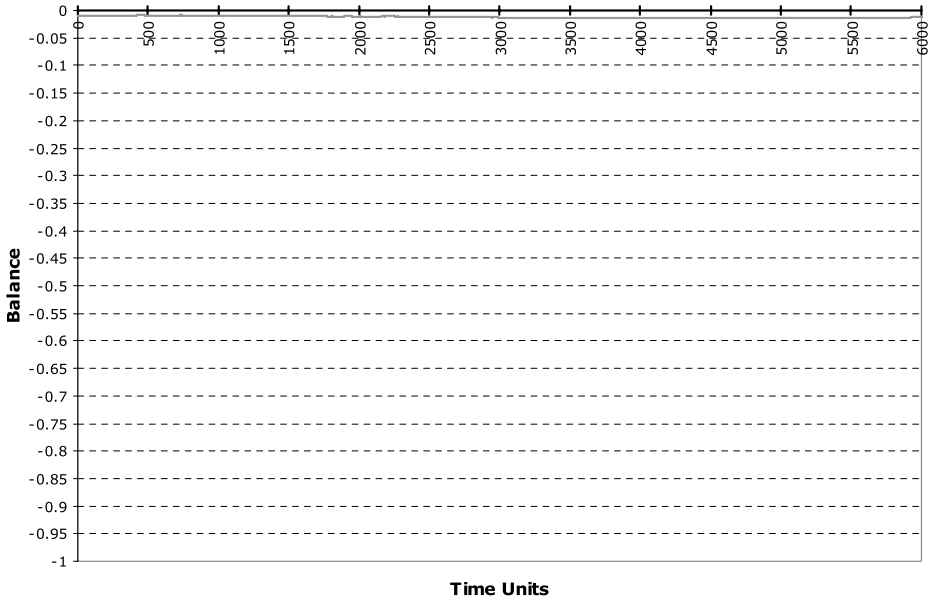
The population will have roughly the same number of "red" and "blue" sensors distributed through the field. It is obvious that sensors are already uniformly distributed and the goal is reached.

The next simulation to run was one involving a particular number of "red" and "blue" sensors randomly (on coordinates) distributed on the field. Figure 10a shows the result of simulation for 15,000 "red" sensors and 5,000 "blue" sensors. Figure 10b 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



(a)

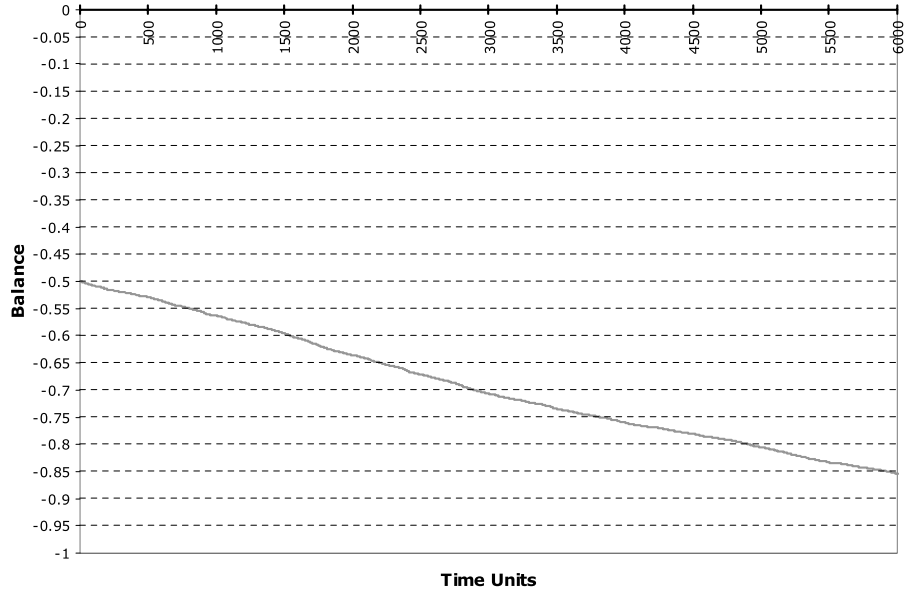


(b)

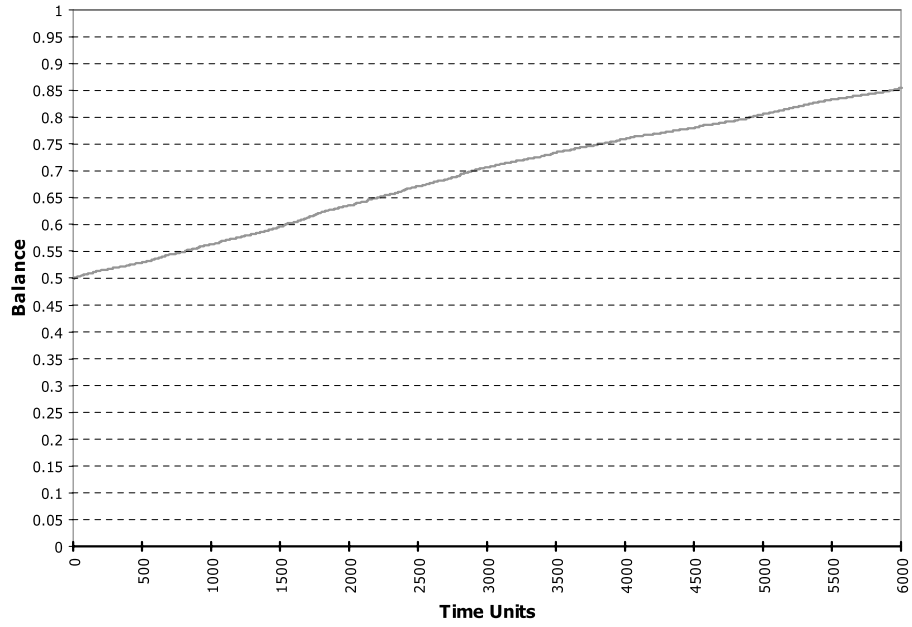
Fig. 9. (a) Randomly deployed sensors; (b) Randomly deployed sensors (different deployment).

"reds" and 5,000 "blues". At the deployment moment there was a serious numeral superiority of "red" sensors (Fig. 10a). 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



(a)



(b)

Fig. 10. (a) 15,000 “red” and 5,000 “blue” sensor nodes deployment; (b) 5,000 “red” and 15,000 “blue” sensors deployment.

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 seats 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

Table 1
Statistics at the deployment moment. Five different experiments

Status	Exp #				
	1	2	3	4	5
sleep	16,768	16,762	16,808	16,716	16,738
L/R	1,057	1,100	1,070	1,111	1,114
L/B	1,096	1,081	1,053	1,101	1,084
transmit	1,079	1,057	1,069	1,072	1,061

Table 2
Red-Blue ratio at the deployment moment

	Total	Reds	Blues
sleep	16,768	12,576	4,192
L/R	1,057	792	265
L/B	1,096	822	274
transmit	1,079	809	270

the longest one. Table 1 shows sensors statistics at the deployment moment.

Furthermore, on a field of 10×10 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 10×10 units there are about 3,300 awake sensors, meaning approximately 33 awake sensors per 1 unit^2. The area of communication region for each sensor is: pi*TR^2 $\approx$ 3.14*1 = **3.14 units**^{2}. Thus, **3.14 * 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:

Using this table it can be derived:

270 transmitting "blue" sensors per 100 units^2 *and each sensor has communication range* (*transmission circle*) *of 3.14 units*^{2} $\Rightarrow$ approximately *8 transmitting "blue" sensors per each sensor's communication range; 809 transmitting "red" sensors per 100 units*^{2} *and each sensor have communication range of 3.14 units*^{2} $\Rightarrow$ 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.

For a better result, it is necessary to solve this collision problem in the following discussion.

Table 3
30 channels utilization.

msgs#	such situations #	percentage	percentage (w/out empty channel)
0	1253905675	63.9052%	–
1	769977566	39.2418%	71.3651%
2	245220150	12.4976%	22.7281%
3	53443461	2.7237%	4.9534%
4	8919375	0.4546%	0.8267%
5	1212660	0.0618%	0.1124%
6	139116	0.0071%	0.0129%
7	13805	0.0007%	0.0013%
8	1139	0.0001%	0.0001%
9	102	0.0000%	0.0000%
10	10	0.0000%	0.0000%
11	1	0.0000%	0.0000%

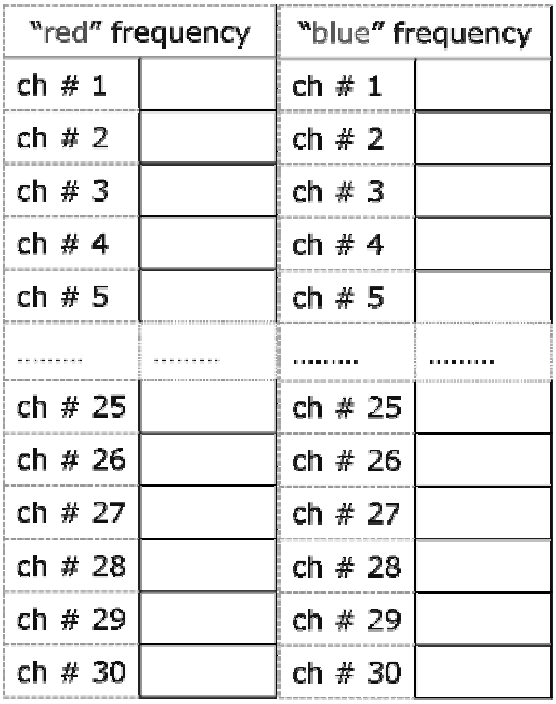
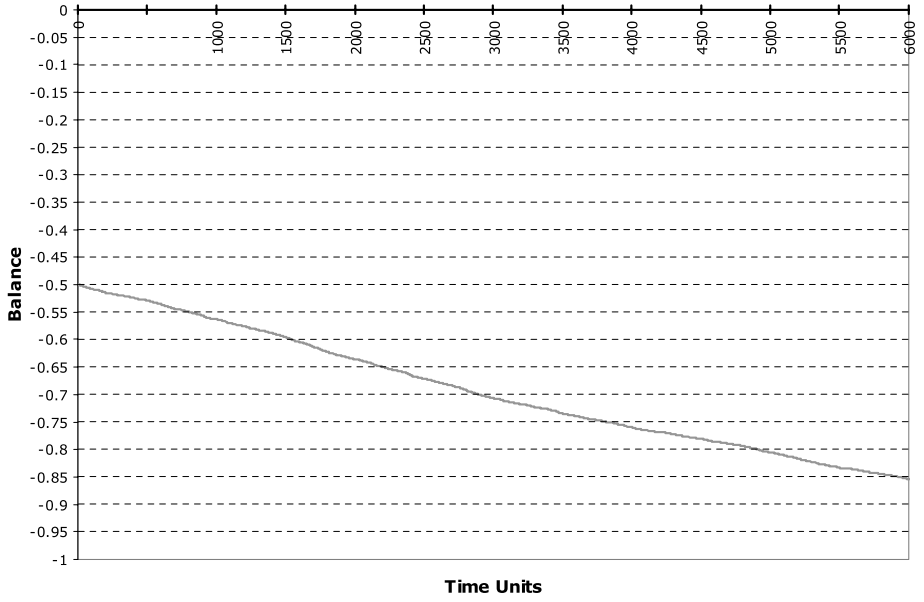


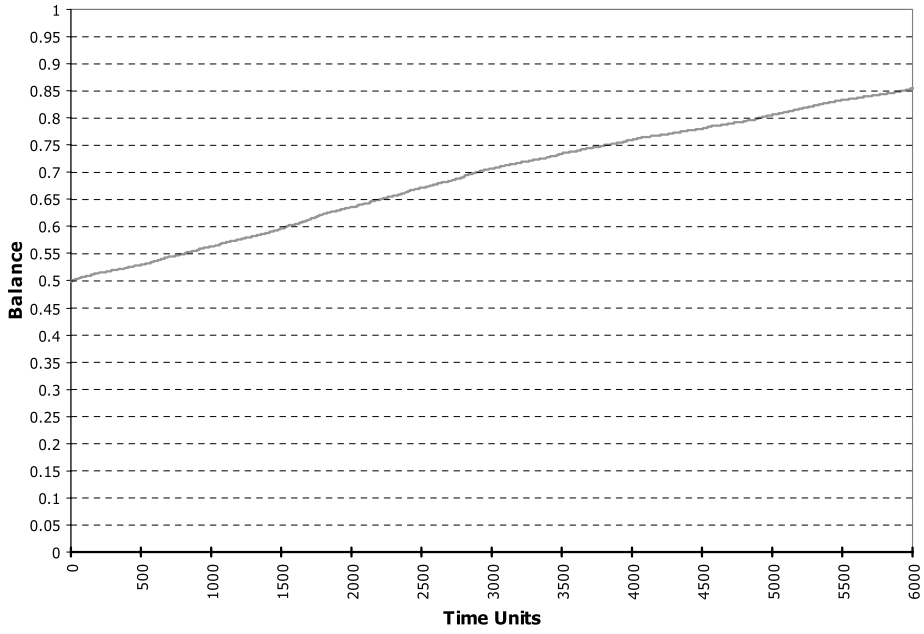
Fig. 11. Frequencies partitioning.

4.3. Channels

During its life cycle every sensor goes through two listening phases: LR and LB phases. They differ from each other by the frequency to which sensor "tunes" its radio receiver: the color of neighbors it needs to listen to corresponds to the "color" of frequency. To reduce the number of collisions each transmission frequency – "red" frequency and "blue" frequency – is partitioned into 30 channels. To transmit - each sensor randomly picks one of 30 channels. To listen to the neighborhood – each sensor scans through all 30 channels of each frequency. If two transmitting nodes meet on the same channel then collision is detected.



(a)

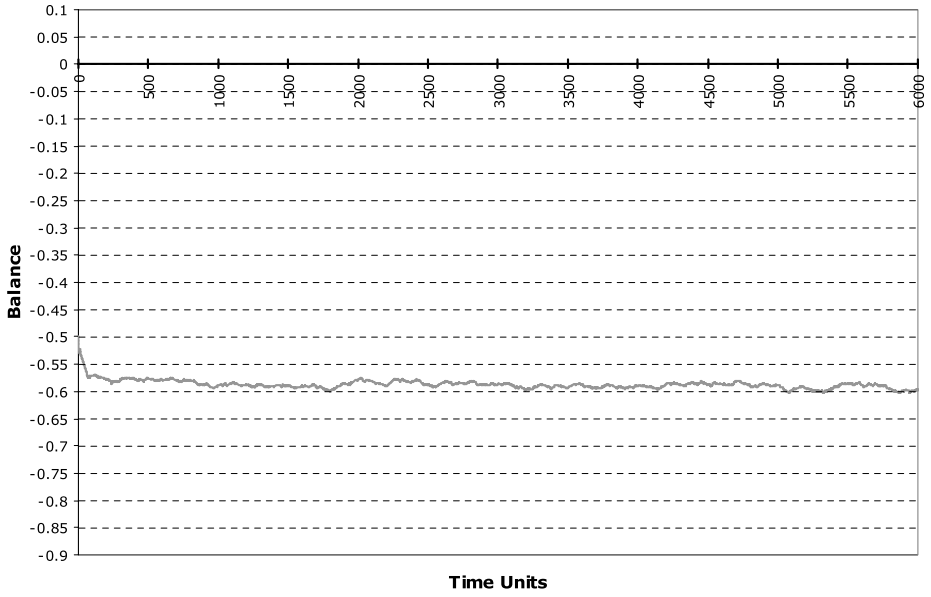


(b)

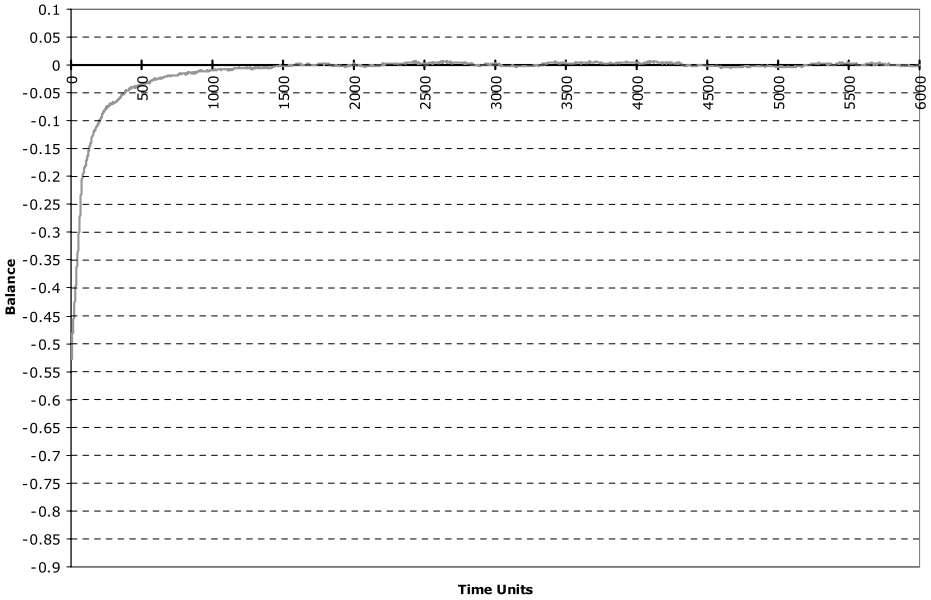
Fig. 12. (a) 15,000 “red” and 5,000 “blue” sensors deployment, 30 channels; (b) 5,000 “red” and 15,000 “blue” sensors deployment, 30 channels.

Figure 12a and 12b show the results of the simulation run with 30 channels.

Since after such model modifications every sensor has a bunch of frequency channels to choose for transmission, the number of collisions is significantly reduced and sensors can hear each other much more



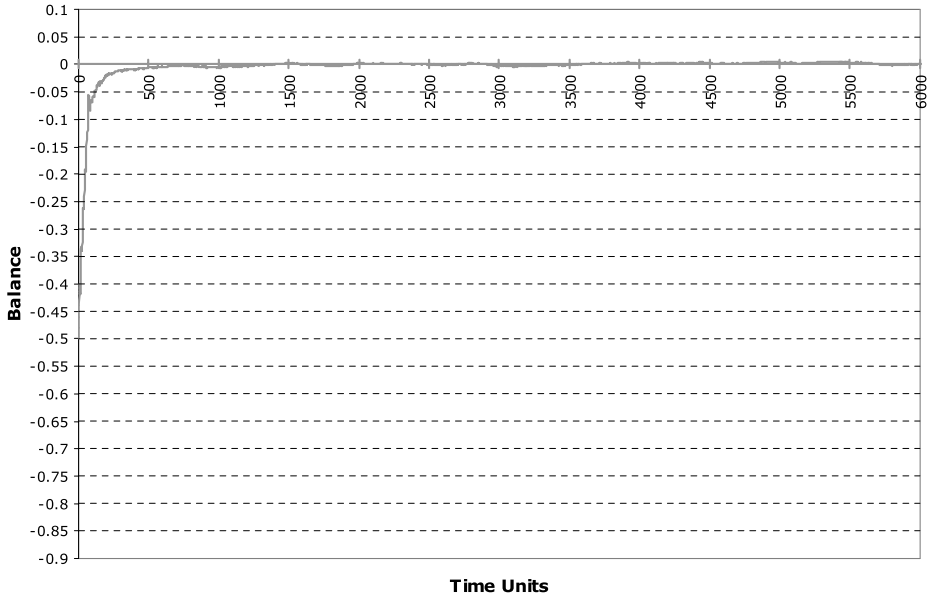
(a)



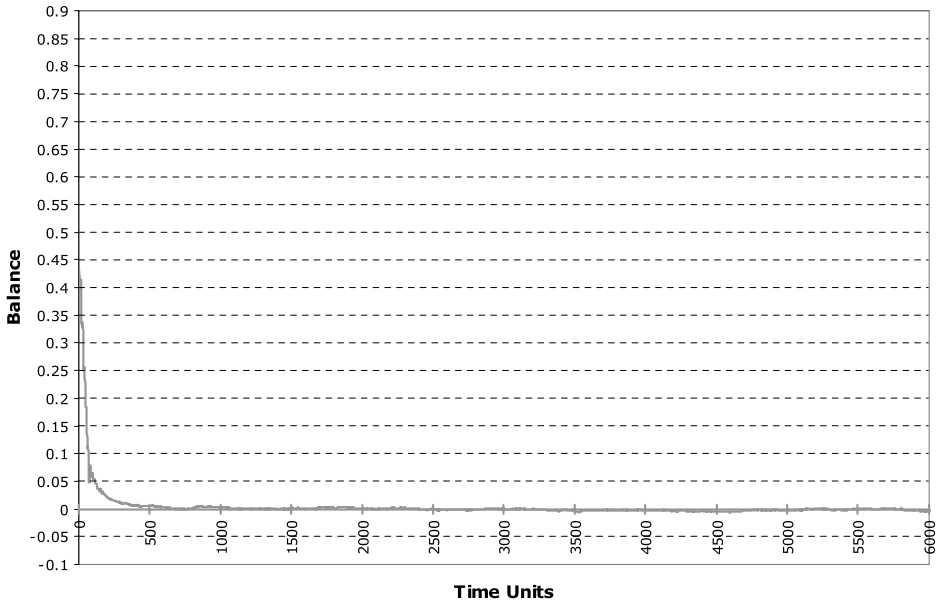
(b)

Fig. 13. (a) 15,000 “red” and 5,000 “blue” sensors deployment, 15 channels; (b) 15,000 “red” and 5,000 “blue” sensors deployment, 45 channel.

clearly. Figure 12a, 12b show big improvement in comparison to the simulation where unpartitioned frequencies were used (Fig. 10a, 10b). During this work it was found experimentally that this number of channels (30 channels) picked for partitioning each of two frequencies is the one which allows getting



(a)

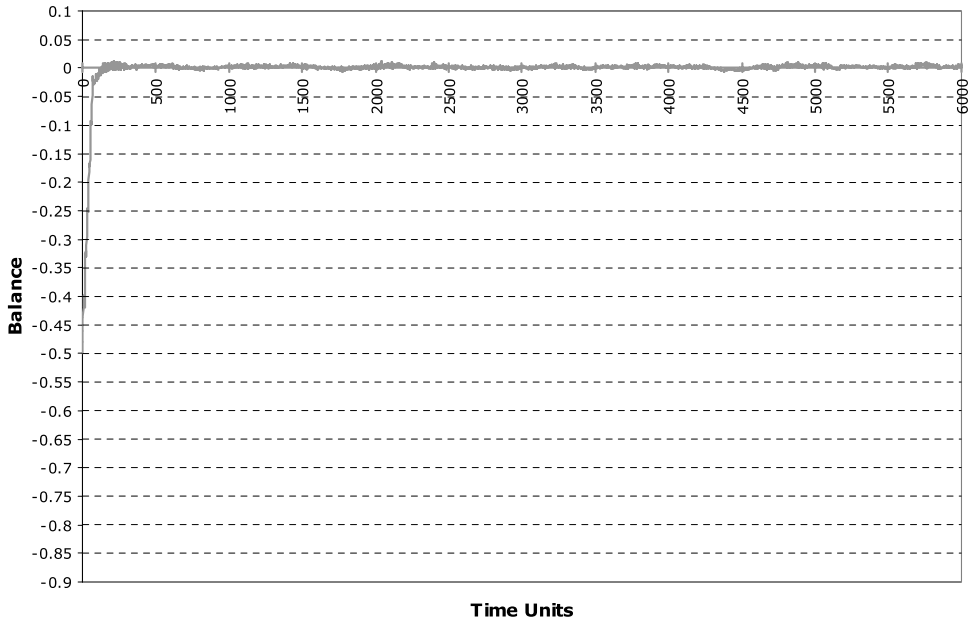


(b)

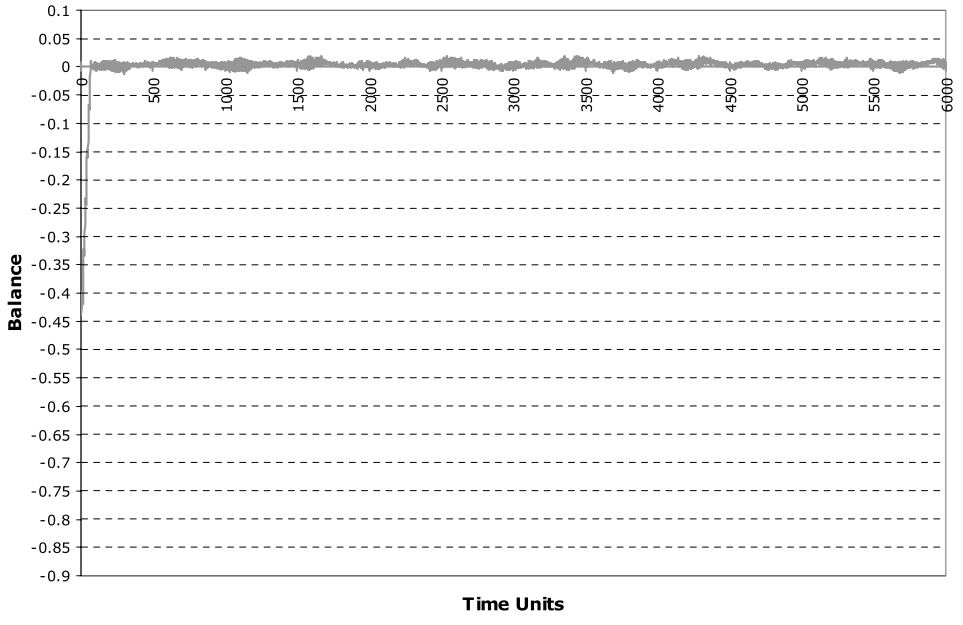
Fig. 14. (a) 15,000 “red” and 5,000 “blue” sensors deployment, 30 channels, Count Collisions; (b) 5,000 “red” and 15,000 “blue” sensors deployment, 30 channels, Count Collisions

optimum results (Fig. 13a, 13b).

Comparing Figs 12 and 14, one can observe an improvement in a speed with which sensors come to the balance.



(a)



(b)

Fig. 15. (a) 15,000 “red” and 5,000 “blue” sensors deployment, conservativity $= \mid 0.10 \mid$; (b) 15,000 “red” and 5,000 “blue” sensors deployment, conservativity $= 0$.

4.4. Influence of the individual sensor conservativity on the emerging network behaviour

Sensor node *conservativity* mentioned in Section 3.5 is a parameter which prevents sensor from a premature decision of changing color. In some sense conservativity is a tolerance to the changing environment from the sensor's point of view. Sensor node sits in the middle of its communication range listening to the neighbors and it has some mechanism which helps it against thoughtless behavior – jumping back and forth between red and blue colors. Thus, conservativity gives the sensor some time to wait and see what will happen further. If the communal behavior goes out of the "threshold", then it is time for the sensor to change its stand – color.

It would be interesting to see how sensors network behavior depends on the individual node conservativity. Refer to Fig. 15, one can make a conclusion that the smaller the conservativity is the bigger the sensor population speed of coming to the balance, but also the bigger is the oscillation between "red" and "blue" sides of the X axis.

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. Channels utilization helped to discover this.

It was shown that the large group of sensor nodes demonstrates an emerging behavior, just like living cells. Each sensor gathers the information from its neighborhood using short-range radio, and makes its decisions based on this local information, the whole network behavior depends on such local decisions. While single node tries to maintain local coloring scheme stabilization, it also contributes to the global red-blue colors stabilization.

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.

Appendix A

Table 4
15 channels utilization.

msgs#	such situations #	percentage	percentage (w/out empty channel)
0	85369914	43.368700%	
1	56059131	28.478500%	50.287600%
2	30129942	15.306300%	27.028000%
3	15190740	7.717030%	13.626800%
4	6624892	3.365500%	5.942830%
5	2436336	1.237680%	2.185510%
6	764007	0.388122%	0.685350%
7	208720	0.106032%	0.187231%
8	49975	0.025388%	0.044830%
9	10732	0.005452%	0.009627%
10	2043	0.001038%	0.001833%
11	384	0.000195%	0.000344%
12	63	0.000032%	0.000057%

Table 5
45 channels utilization.

msgs#	such situations #	percentage	percentage (w/out empty channel)
0	416762436	70.560600%	–
1	144575369	24.477600%	83.145600%
2	25855379	4.377480%	14.869500%
3	3138065	0.531295%	1.804710%
4	290152	0.049125%	0.166867%
5	21849	0.003699%	0.012565%
6	1357	0.000230%	0.000780%
7	58	0.000010%	0.000033%

References

[1] D. Gračanin, M. Eltoweissy, S. Olariu, A. Wadaa, Extensible Service-Centric Model for Wireless Sensor Networks, Journal of Wireless Networks, Kluwer Academic Publishers, 2005.
[2] D.M. Doolin and N. Sitar, Wireless Sensors for Wild Monitoring, *Proc SPIE Symposium on Smart Structures & Materials* (*NDE 2005*), San Diego, California, March 6–10, 2005.
[3] E. Bonabeau, M. Dorigo and G. Theraulaz, Inspiration for Optimization from Social Insect Behaviour, *Nature* **406** (July 6 2000), 39.
[4] I.F. Akyldiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Wireless Sensor Networks: A Survey, *Computer Networks* **3894** (2002), 393–422.
[5] J. Epstein, Learning To Be Thoughtless: Social Norms and Individual Computation, Center on Social and Economic Dynamics Working Paper No. 6, revised January, *Forthcoming in Computational Economics*, 2000.
[6] J. Packard, in press, Social Behavior of Wolves: Reproduction and Development in Family Groups, in: *The Ecology and Behavior of the Wolf*, The University of Chicago Press, L.D. Mech and L. Boitani, eds, Chicago.
[7] J. Dornstetter, D. Krob, M. Morvan and L. Viennot, Some Algorithms for Synchronizing Clocks of Base Transceiver Stations in a Cellular Network, *Journal of Parallel and Distributed Computing*, 1999.
[8] J.M. Epstein and R.L. Axtell, Growing Artificial Societies: Social Science from Bottom Up, The MIT, Press, 1996.
[9] J.M. Kahn, R.H. Katz and K.S.J. Pister, Next Century Challenges: Mobile support for Smart Dust, *Proc ACM MOBICOM* Seatle, WA, (August 1999), 271–278.
[10] K. Jones, K. Lodding, S. Olariu, A. Wadaa, L. Wilson and M. Eltoweissy, Biomimetic Models for Sensor Networks - Towards a Social Sensor Network, Handbook of Bio-Inspired Algorithmic Techniques, CRC Press, Boca Raton, 2005.

[11] K. Jones, K. Lodding, S. Olariu, L. Wilson and C. Xin, *Sensor Networks for Situation Management: A Biomimetic Model*, http://ecaaser3.ecaa.ntu.edu.tw.
[12] Kelly, Out of Control: The New Biology of Machines, Social Systems and the Economic World, Perseus, Books Group, 1995.
[13] A. Mainwaring, J. Polastre, R. Szewczyk and D. Culler, Wireless Sensor Networks for Habitat Monitoring, Intel research, IRB-TR-02-006, June 10, 2002, 2002 ACM International Workshop on Wireless Sensor Networks and Applications. Retrieved April5, 2005 from http://www.greatduckisland.net.
[14] O. Holland and C. Melhuish, Stigmergy, Self- Organization and Sorting in Collective Robotics, *Artificial Life* **5**(2) (1999), 173–202, MIT Press.
[15] http://www.ai-junkie.com/competition.htm.
[16] R. Ishak, S. Salleh, S. Olariu and M.I.A. Aziz, SPLAI: Computational Finite Element Model for Sensor Networks, *International Journal of Mobile Information System* (*IJMIS*) **2**(1) (2006), 77–92, IOS Press.
[17] R. Ishak, Q. Xu, S. Olariu and S. Salleh, Hybrid Training with Binary Search Protocol for Wireless Sensor Networks, *International Journal of Mobile Information Systems* **3**(3–4) (2007), 233–249, IOS Press.
[18] R. Ishak, S. Olariu and S. Salleh, Multi-training Sensor Networks for Bipartite Conflict Graphs. The ACM International Workshop in Middleware for sensor networks, *Melbourne* Australia, 27 Nov- 1 December, 2006.

Ruzana Ishak is a senior Lecturer of Science Department at Univ. Teknologi Malaysia *CityCampus*, Kuala Lumpur. She received the B.Sc degree in Computational Maths and CAGD from Univ. Sains Malaysia, Penang in 1993 and the M.Sc degree in Mathematics from Univ. Teknologi Malaysia, Johor Bahru in 2002. She received her Phd in 2007 from the same university on Multi-training techniques for localization in wireless sensor networks. Her research interests are in the area of graph theory application models, wireless sensor network location systems and mobile computing. Email:ruzanaishak@yahoo.com

Stephan Olariu is a tenured full professor in Computer Science at Old Dominion University. He is a world-renowned technologist in the areas of parallel and distributed systems, parallel and distributed architectures and networks. He was invited and visited more than 120 universities and research institutes around the world lecturing on topics ranging from parallel algorithms, to graph theory, to wireless networks and mobile computing, to biology-inspired algorithms and applications, to telemedicine, to wireless location systems, and sensor network applications. Professor Olariu is the Director of the Sensor Networks Research Group at Old Dominion University. He has published 200+ archival journal articles and 100+ conference papers. Professor Olariu earned his BSc, MSc and Ph.D. (Computer Science) at the McGill University, Montreal, Canada. He has received an NSF Research Initiation award. Stephan is an Associate Editor of "Networks" And and serves on the editorial board of "IEEE Transactions on Parallel and Distributed Systems" and "Journal of Parallel and Distributed Computing". Prof. Olariu's current research interests are in the area of parallel and distributed systems, wireless networks performance evaluation and security. Email:olariu@cs.odu.edu

Shaharuddin Salleh is Professor in Computational Mathematics at the Department of Mathematics, Universiti Teknologi Malaysia. He obtained PHD degree in 1998 from the same university. Prof Salleh has published 4 books and 58 technical papers. His research interests include numerical modeling and simulation, parallel algorithms, graph theoretical applications and mobile computing. Email: ss@mel.fs.utm.my