2013 13th International Conference on Control, Automation and Systems (ICCAS 2013)
Oct. 20-23, 2013 in Kimdaejung Convention Center, Gwangju, Korea

A CA Model for Target Tracking in Distributed Mobile Wireless Sensor Network

Sang-Ki Ko, Hwee Kim and Yo-Sub Han*

Department of Computer Science, Yonsei University, Seoul 120-749, Korea
(Tel : +82-2-2123-7434; E-mail: {narame7,kimhwee,emmous}@cs.yonsei.ac.kr)
* Corresponding author

Abstract: We present a cellular automaton model that efficiently tracks a target in distributed mobile wireless sensor network. We focus on simulating the mobile wireless sensor networks for tracking a moving target considering energy-efficiency. Using the mobility of sensors, we first disperse sensors as far as possible for the better coverage while preserving connectivity. Furthermore, we introduce a tri-level alert system for activating sensors selectively. We conduct experiments for evaluating the performance of the proposed model. The performance of our model is measured in terms of lifetime, coverage and tracking quality, and compared with other models.

Keywords: Cellular Automata, Mobile Wireless Sensor Networks, Target Tracking, Distributed Algorithms

1. INTRODUCTION

Wireless sensor networks (WSNs) play important roles in a wide range of applications such as industrial control and monitoring [14], security [15], military sensing [9]. A wireless sensor usually consists of a processor, memory unit and power supply, and is therefore limited in its processing power, memory and transmission range. A sensor can sense, measure and gather information from the environment. It can also communicate with other sensors, but only within its limited range. A WSN has a limited lifespan since sensor batteries offer finite energy reserves. Many researchers focus on the maximization of WSN lifespan and coverage [3], [8]. The coverage problem arises from the finite nature of batteries. Especially, we consider the WSNs that consist of mobile sensors. We call such WSNs mobile wireless sensor networks (MWSNs). The key difference is the mobile nodes can reposition themselves and organize the topology of the network. Due to the mobility of sensors, MWSN is more capable than WSN, and needs more considerations to design an efficient protocol.

We simulate MWSNs for tracking a mobile target using cellular automata (CAs). A CA consists of a regular grid of cells, each of which is in one of a finite number of states. Each cell has a state and a set of neighboring cells called *neighbors* around itself. The state of each cell changes depending on the states of neighboring cells. Many researchers have focused on applications of cellular automata to address phenomena caused by local interactions.

We propose a CA model that simulates MWSNs for target tracking problem. We consider several measurements for evaluating the proposed model: lifetime, coverage and tracking quality. In Section 2, we outline the basic concepts of cellular automata and describe the existing cellular automaton models simulating WSNs and MWSNs. In Section 3, we introduce the target tracking problem and the related works for the problem. We present our model in Section 4. We provide simulation results in Section 5.

2. CA MODELS FOR WSNS

A CA is formally specified by a quadruple (L, S, N, f), where L is a regular grid composed of cells, S is a finite set of states, N is a finite set of neighborhood indices, such that $\forall c \in N$ and $\forall r \in L : r + c \in L$ and $f : S^n \rightarrow S$ is a transition function [16]. A configuration $C_t : L \rightarrow S$ is a function that associates a state with a cell of the grid L. The configuration of the next time step is defined as $C_{t+1}(r) = f(C_t(i)|i \in N(r)))$, where $N(r)$ is a set of all neighbors of the cell r. In other words, every cell updates its state based on the current states of neighboring cells. Cellular automata can be applied to the simulations of WSNs, as we can find similarity between a CA and a WSN. Every sensor in a WSN can communicate with sensors located within the communication range of the sensors. In particular, we consider the homogeneous WSN where every sensor in the network is identical. Also, we assume that the sensors are mobile and work in a fully distributed manner. Each sensor moves and changes its state according to the states and positions of the neighboring sensors. Since the sensors are unaware of the states of sensors that are not in neighboring cells, they should transfer the important events such as the target detection to their neighbors.

WSNs are employed in various applications [9], [14], [15]. Before we establish a WSN for a practical purpose, we first design a protocol for the new WSN and evaluate the proposed protocol. Since it is very expensive to establish a WSN and run a field test for evaluative purposes, WSNs are typically evaluated by simulation tools such as NS [1] or SensorSim [12]. Based on the simulation results, we can estimate an expected performance of the proposed protocol and modify the protocol if neces-

sary.

Researchers have already used CAs for WSN simulations [6], [7], [11]. Especially, two-dimensional CAs have interesting properties that are suitable for simulating homogeneous WSNs. Cunha et al. [7] developed a WSN simulator based on CAs called CASim. They implemented a topology control algorithm in CASim to perform node scheduling. Choudhury et al. [6] extended the CA algorithms in CASim by designing improved node scheduling algorithms. They also proposed CA algorithms for optimizing MWSNs [4], [5]. For randomly deployed sensors, they aimed to disperse sensors as far as possible for maximizing the MWSN coverage [4]. Later, they studied the same problem while maintaining the connectivity between mobile sensors [5].

Now we formally define our CA model. We have a CA $A = (L, S, N, f)$ for simulating MWSNs. Each cell $c_i \in L$ has a composite state $s_i \in S$ for indicating the state of a sensor. We define S to be a set of composite states, where the composite states are defined as combinations of sub-states described in Table 1.

Table 1: Descriptions of the cell states in our CA model

State	Notation	Description
Existence	$b_i \in B$	If there is a sensor in c_i, b_i is 1, otherwise, 0.
Activity	$a_i \in A$	If $b_i = 0$, $a_i = 0$. If $b_i = 1$, a_i is one of the following three states: active ($a_i = 1$), stand-by ($a_i = 2$), and dead ($a_i = 3$)
Alerting state	$p_i \in P$	If $b_i = 0$, $p_i = 0$. If $b_1 = 1$, p_i is one of following five states: normal ($p_i = 1$), detected ($p_i = 2$), 1-level alerted ($p_i = 3$), 2-level alerted ($p_i = 4$), final alerted ($p_i = 5$), and miss alerted ($p_i = 6$).
Energy	$e_i \in E$	The remaining energy of a sensor c_i. The unit of energy is the Joule.

As shown in Table 1, the composite state s_i of a cell c_i represents the current state of a sensor that is possibly in the cell in the time step. Formally, we define the set of states as $S = B \times A \times P \times E$. The state of a cell c_i is represented by a 4-tuple $s_i = (b_i, a_i, p_i, e_i)$. Now we explain the meaning of each sub-state. Since the grid of the CA is assumed as a certain area, we can assume that each cell means a sub-area. We represent whether or not there exists a sensor in the sub-area of c_i by the sub-state b_i. If there is a sensor in c_i, we denote the sensor by $\mathtt{s}(\mathtt{c_i})$. There are two types of sensors in the grid. Active sensors are sensing their environment within the sensing radius R_s while stand-by sensors are inactive and not sensing. Both types of sensors communicate with other sensors within the communication radius R_c. Sensors move into a dead state when they run out of energy. Once a sensor moves into a dead state, the sensor does not sense and communicate anymore. These states are represented by the sub-state a_i. The sensors in our algorithm has another state called the *alerting state*. According to the level of the alert message arriving the sensor in c_i, the alerting state of the sensor changes. The remaining energy means the amount of the energy that remains in the battery of the sensor.

3. TARGET TRACKING PROBLEM

Many researchers studied the target tracking problem in WSNs [2], [10], [13]. The problem is to detect an object in the network area and track the moving path of the object. We call the detected object *target*. Pattem et al. [13] studied the energy-quality trade-offs of the several strategies for sensor activation. They compared the performance of the following activation strategies:

- Naive activation (NA),
- Randomized activation (RA),
- Selective activation based on prediction (SA),
- Duty-cycled activation (DA).

In NA, all sensors are active all the time. This implies that sensors can always detect or track the object within R_s. In RA, each sensor is active with a probability p. SA forces only a small subset of sensors to be active by prediction at any given point of time. DA periodically forces sensors to be active and stand-by with a regular duty cycle. They showed that SA is dominating among all strategies. Furthermore, in conjunction with DA, SA saves more energy with the small decrease in quality.

Our problem is to design CA algorithms for MWSNs that are specially designed for *target tracking problem*. Here we assume that mobile sensors in the network are equipped with binary detectors. Note that binary detectors can provide only one bit of data indicating the presence or absence of a target in the sensing radius. Our algorithms aim to detect and track a single object (target) in the network.

We consider the scenario that all mobile sensors are deployed from the central base station. They disperse from the base station as far as possible to detect the object while preserving the connectivity with other sensors. Note that preserving connectivity is a very important issue in the target tracking problem since the sensors should handover the tracking results to the base station.

4. CA MODEL FOR THE TARGET TRACKING PROBLEM

We have a two-dimensional (2D) CA $A = (L, S, N, f)$ for simulating MWSNs for the target tracking problem. We design a movement rule of a sensor based on the radius 1 Moore neighborhood. Assume that there is a sensor $\mathtt{s}(\mathtt{c})$ on the central cell c. Then, $\mathtt{s}(\mathtt{c})$

can move any of nine cells within radius 1. If $\mathtt{s}(\mathtt{c})$ moves out of the cell c and no other sensor comes in, then b_i becomes 0.

4.1 Movement Rules

Basically, our rules for sensor movement are based on the movement rule of Choudhury et al. [4]. The major concern of their rules is to disperse sensors for maximizing the coverage of MWSN while maintaining connectivity. According to our scenario, sensors start from the base station located at the center. We also should disperse the sensors for the better coverage since the coverage is a very important measurement for tracking quality. However, we should more carefully consider the possibility of connectivity breaks, because we may lose the target due to the connectivity breaks.

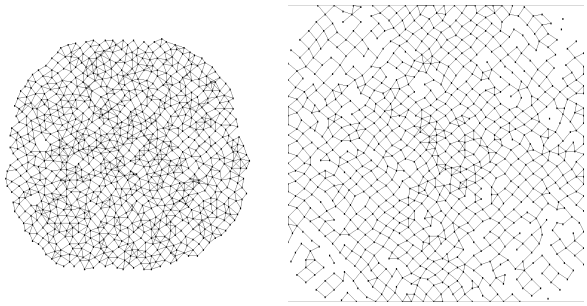
We consider the states of $(2 \cdot R_c + 1)^2 - 1$ cells around a sensor to determine the movement of the sensor. Because some of cells are empty, we consider at most $(2 \cdot R_c + 1)^2 - 1$ sensors around the sensor. First, we determine the movement of a sensor in the x-direction. The weights are assigned to the sensors within R_c. For the sensors at distance d in the x-direction, we assign $R_c - d$. Next we calculate the sum of weights of sensors in the positive and negative x-direction, separately. Suppose w_+ and w_- are the values. Then, the movement of the sensor in the x-direction is determined as follows:

- No move: if $-k < w_- - w_+ < k$,
- Left move: if $w_- - w_+ < -k$, and
- Right move: if $w_- - w_+ > k$,

where k is a threshold. We can determine the movement in the y-direction analogously. This rule just disperses sensors from each other and breaks the connectivity between sensors at last. For this reason, we designed an additional rule called the *neighbor count rule*. Given two thresholds k_1, k_2 where $k_1 < k_2$, a sensor does not move if $k_1 \leq n \leq k_2$, where n is the number of sensors located within R_c from the cell. This prevents the sensors on the boundary of a MWSN from further movements. Fig. 1 shows the results when we employ the neighbor count rule with $k_1 = 3$ and $k_2 = 5$. Although the coverage of the network is smaller than the case without the rule, the sensors still maintain a high connectivity after the dispersion. The important thing to note is the boundary of the MWSN. In Fig. 1(a), all sensors even at the boundary are connected to at least two other sensors. This is very important when the target goes out of the boundary and the MWSN loses the target. We explain the reason when we discuss the problem of handling lost targets in Section 4.4

4.2 Rules for Energy-Efficiency

As mentioned in Section 3, energy-efficiency is one of the significant issues in a MWSN for target tracking. Therefore, we need to control the states of the sensors in a MWSN to decrease energy consumption. The sensor activation strategy is used for this problem and Pattem et al. [13] studied several generic activation strategies and compared the performance. Clearly, it seems that selective activation with prediction is the best strategy for sensor activation problem when there is a target in a MWSN. However, the strategy does not consider the case when there is no object in the MWSN or the MWSN loses the target during the tracking. This implies that sensors in the MWSN should be active to detect the region where the object can appear.



(a) (b)

Fig. 1: Comparison of two configurations after 500 time steps: a) with the neighbor count rule and b) without the rule.

The MWSN should sense the area as much as possible before the target appears. That means we need to consider the problem of maximizing the coverage of the MWSN while saving energy consumption. Many previous researches on this problem use the information from neighboring sensors [6], [7], [11]. If there are enough number of active sensors around the active sensor, turn the sensor into the stand-by state. Similarly, if there are not enough number of active sensors around the stand-by sensor, turn the sensor into the active state. We adopt a similar approach for this problem. Our sensor activation strategy is as follows:

- If a sensor in the active state has more than n_1 active neighbors, then it should move into the stand-by state;
- If a sensor in the stand-by state has less than n_2 active neighbors, then it should move into the active state.

This sensor activation rule is more energy-efficient than naive activation rule since some of sensors are in the stand-by state by the rule. Furthermore, the MWSN following our activation rule still covers enough area. We save the extra energy consumption wasted from the overlaps of sensing area between sensors. Our rule randomly applies to all sensors whose alerting states are “normal”.

4.3 Tri-Level Alerting System

The *tri-level alerting system* is the main idea of our model. When a target is detected by some sensors in the MWSN, the sensors deliver the information to all other sensors by this system. If the MWSN has a central base station broadcasting the information, the problem may be simpler because we can efficiently control the sensors by

the base station. However, the problem is more complicated since we study a fully distributed MWSN that has no base station. The CA rules for the tri-level alerting system are as follows:

- Initially, the alerting states of all sensors begin with "normal" states;
- If a sensor detects a target by sensing environment, then the alerting state of the sensor turns into "detected";
- If a sensor has any sensor whose alerting state is "detected", then the alerting state of the sensor changes into "1-level alerted";
- If a sensor has any sensor whose alerting state is "1-level alerted", then the alerting state of the sensor changes into "2-level alerted";
- If a sensor has any sensor whose alerting state is "2-level alerted", then the alerting state of the sensor changes into "final alerted".

These rules are to notify that the target is detected by some sensors. The alerting states are used to activate or inactivate the sensors to save energy consumption. We change the activity of sensors according to the alerting states. We set the states of sensors to active for sensors whose alerting states are "detected", "1-level alerted", or "2-level alerted". On the other hand, we set the states of sensors to stand-by for sensors whose alerting states are "final alerted". Intuitively, we force the sensors who detected the target and the sensors around them to be active and otherwise, force the sensors to be stand-by. Then, the MWSN can track the moving target in an energy-efficient way since only a small subset of active sensors track the target while other stand-by sensors save their energy. See Fig. 2 for example.

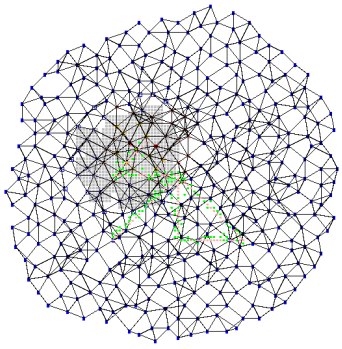


Fig. 2: The gray region means the area being sensed by sensors activated by the tri-level alerting system. The other sensors are "final alerted", and thus in the stand-by state.

4.4 Rules for Handling Lost Target

The proposed model should be able to manage the situation when a MWSN lose the target. Note that the tri-level alerting system only activates a subset of sensors around the detected target for the energy-efficiency of the MWSN. Therefore, when the target is lost during the target tracking phase, all sensors are forced to be in stand-by states as they are "final alerted" by other sensors. Then, the MWSN does not sense the environment, and thus cannot detect the target anymore. This is the reason why we need additional rules for handling lost target.

Now we introduce rules for handling this situation. When a MWSN lose a target, the sensors who are alerted most recently should broadcast the loss of the target to all of sensors in the MWSN to activates them. This is implemented as the "miss alerted" state in our CA rule. The rules are as follows:

- The rules only apply to the sensors whose current alerting state is "detected" and changes into other states in the next time step,
- If there is no neighboring sensors whose alerting state is "detected", the alerting state of the sensor changes into "miss alerted" in the next time step.

Note that once any sensor changes into the "miss alerted" state, the alerting state does not change by the other sensors. We assume that each sensor contains an own counter and begins counting the time steps after it becomes "miss alerted". After a certain amount of time steps, these sensors turn into the "normal" state.

4.5 Rule Hierarchy

Now we review the proposed rules and introduce an integrated verification rule for a sensor to determine the next state in the CA. The proposed rules have hierarchy to be applied to the sensors. We introduce the rules in order of the hierarchy of rules.

1. If an activate sensor detects the target, then the alerting state turns into "detected".
2. A sensor in "miss alerted" state only changes the alerting state by the own counter of the sensor.
3. If any neighboring sensor is in the "detected" state, then the alerting state turns into "1-level alerted".
4. If any neighboring sensor is in the "1-level alerted" state, then the alerting state turns into "2-level alerted".
5. If any neighboring sensor is in the "miss alerted" state and the alerting state is not "normal", then the alerting state turns into "miss alerted".
6. If any neighboring sensor is in the "2-level alerted" state, then the alerting state turns into "final alerted".

The important thing to note is the hierarchy between the rules for the tri-level alerting system and the rules for handling lost target. When the target is lost, the sensors deliver the information with the "miss alerted" state prior to the rules for the tri-level alerting system. Once all sensors in the MWSN turns into "miss alerted" states, the alerting states of them do not change by the other sensors and turns into the "normal" state by individual counters.

5. SIMULATION RESULTS

We present the simulation results of our model compared to the other basic strategies. The strategies used for comparisons are the naive activation and the randomized activation strategies. We also compared with the strategy only using the energy-efficient activation rule and called the strategy Count.

We provide the experimental set-up of our simulations. We deploy 400 sensors on the center of the grid. Each sensor can sense the area where the distance is up to 3 and communicate with the sensors at the distance up to 6. Simply, $R_c = 6$ and $R_s = 3$. The sensor initially has a battery of 5J and consumes 0.165J when active and 0.0006J when stand-by. We set two parameters k_1 and k_2 for the neighbor count rule to 3 and 5, respectively.

Fig. 3 shows the average coverage of the given four strategies including our model. Because we randomly activates the sensors with 50%, the results of the naive and randomized strategies are reasonable. The coverage of the Count strategy remains much longer than the naive and randomized strategy as we expected. The coverage of our strategy is irregular due to the moving target. We remind that in our strategy, if the MWSN lose the target, all sensors turn into the "miss alerted" state and reactivated following the energy-efficient rule.

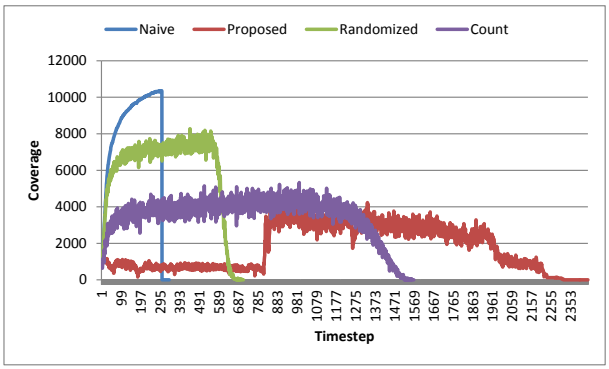


Fig. 3: The graph shows the average coverage of strategies. The positive x-direction means the flow of the discrete time step.

Fig. 4 shows the total residual energy of the MWSN. That means the sum of residual energy in all sensors. This figure indicates that our strategy remarkably saves much more energy than other strategies.

Table 2 compares the simulation results of four strategies in terms of lifetime, tracking quality, and tracking success rate. Our strategy shows the longest lifetime than other strategies. The Count strategy shows longer lifetime than the randomized strategy. The naive strategy is the worst strategy when it comes to the lifetime. Tracking error rate is defined as the average difference between the estimated location by the sensors and the actual location of the target. Tracking success rate is measured with the first 300 time steps of MWSNs since the average lifetime of the naive strategy is slightly over 300. Since we

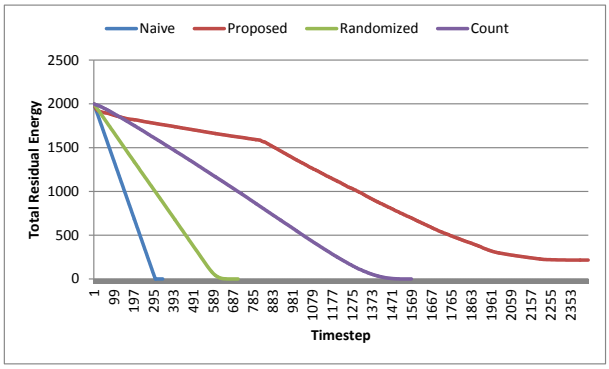


Fig. 4: The graph shows the average residual energy of strategies. The positive x-direction means the flow of the discrete time step.

only compares the first 300 time steps, the naive strategy shows the best result among four strategies. Even though our model is slightly worse than the naive one, the lifetime is almost seven times longer than the naive strategy. The randomized and the Count strategies show the trade-off between the lifetime and the tracking quality of the MWSN.

Table 2: The table shows the comparisons of four strategies including the proposed model. We compared the strategies in terms of lifetime, tracking error (TE), and tracking success rate (TSR).

Strategy	Naive	Random	Count	Proposed
Lifetime	302.54	704.52	1523.54	2512.67
TE	1.09	1.78	2.14	1.11
TSR (%)	100	76.85	39.60	99.66

6. CONCLUSIONS

We have introduced a cellular automaton model for the distributed MWSN. Especially, we have considered the specialized MWSN for the target tracking problem. We have proposed the tri-level alerting system and additional rules for handling lost target. Our model exhibits relatively longer lifetime than other basic strategies while maintaining high tracking quality.

In future, we take into consideration the energy consumption from the movements of sensors. By designing the movements of sensors more efficiently, we can provide a better CA model for MWSNs.

ACKNOWLEDGEMENT

This research was supported by the Basic Science Research Program through NRF funded by MEST (2010-0009168).

REFERENCES

[1] The Network Simulator ns-2 (v2.35). `http://www.isi.edu/nsnam/ns`, 2011.

[2] S. Bhatti and J. Xu. Survey of target tracking protocols using wireless sensor network. In *Proceedings of the Fifth International Conference on Wireless and Mobile Communications*, pages 110 –115, 2009.

[3] M. Cardei and J. Wu. Energy-efficient coverage problems in wireless ad-hoc sensor networks. *Computer Communications*, 29(4):413–420, 2006.

[4] S. Choudhury, S. G. Akl, and K. Salomaa. Energy efficient cellular automaton based algorithms for mobile wireless sensor networks. In *Proceedings of the 2012 IEEE Wireless Communications and Networking Conference*, WCNC'12, pages 2341–2346, 2012.

[5] S. Choudhury, K. Salomaa, and S. G. Akl. Cellular automaton based motion planning algorithms for mobile sensor networks. In *Procceedings of the 1st International Conference on Theory and Practice of Natural Computing*, TPNC'12, pages 108–120, 2012.

[6] S. Choudhury, K. Salomaa, and S. G. Akl. A cellular automaton model for wireless sensor networks. *Journal of Celluar Automata*, 7(3):223–241, 2012.

[7] R. O. Cunha, A. P. Silva, A. A. F. Loureiro, and L. B. Ruiz. Simulating large wireless sensor networks using cellular automata. In *Proceedings of the 38th annual Symposium on Simulation*, pages 323–330, 2005.

[8] C.-F. Huang and Y.-C. Tseng. The coverage problem in a wireless sensor network. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 115–121, 2003.

[9] M. Hussain, P. Khan, and K.-S. Kwak. WSN research activities for military application. In *Proceedings of the 11th International Conference on Advanced Communication Technology*, pages 271–274, 2009.

[10] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham. On target tracking with binary proximity sensors. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, 2005.

[11] W. Li, A. Y. Zomaya, and A. Al-Jumaily. Cellular automata based models of wireless sensor networks. In *Proceedings of the 7th ACM international symposium on Mobility management and wireless access*, pages 1–6, 2009.

[12] S. Park, A. Savvides, and M. B. Srivastava. SensorSim: a simulation framework for sensor networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 104–111, 2000.

[13] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *Proceedings of the 2nd international conference on Information processing in sensor networks*, IPSN'03, pages 32–46, 2003.

[14] X. Shen, Z. Wang, and Y. Sun. Wireless sensor networks for industrial applications. In *Proceedings of the Fifth World Congress on Intelligent Control and Automation*, pages 3636–3640, 2004.

[15] Y. Wang, G. Attebury, and B. Ramamurthy. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 8(2):2–3, 2006.

[16] S. Wolfram. *A new kind of science*. Wolfram Media, Champaign, Ill, 2002.