정확도와 비용간의 tradeoff를 고려한 이벤트 감지 분산 알고리즘
(Distributed Event Detection Algorithm considering Tradeoff between Accuracy and Cost)

나 현 숙 [†] 녀 뚜 안 안 [††]
(Hyeon-Suk Na) (Nhu Tuan Anh)

요 약 본 논문에서는 무선 센서 네트워크상의 이벤트 감지 문제를 푸는 동시에, 정확도와 비용간의 trade-off를 조절할 수 있는 이벤트 감지 분산 알고리즘을 제안한다. 제안된 알고리즘은 관심영역에 랜덤하게 분포시킨 많은 수의 센서들로 감시되는 무선 네트워크를 모델로 하며, 센서들은 0/1-센서로서 낮은 성능의 배터리를 가지며, 감시, 통신, 그리고 아주 간단한 사칙연산만을 수행하는 정도의 극히 제한된 기능성을 갖는다.

이벤트 감지 문제에서는 언제나, 이벤트가 일어나는 순간 그것을 감지하고, 주변의 이웃들에게 이를 알려 그 이상의 이벤트들도 감지할 수 있도록 하는 센서들이 필요하다. 총 파워비용을 줄이거나 모니터링 시간을 최대화하기 위한 스케쥴링 방법들은 광범위하게 연구되어 왔다. 우리의 방법론은 다음과 같다. 일정한 밀도로 분포되도록 미리 선택한 센서들-여기서는 critical sensor라고 부름-은 언제나 깨어있도록 하고, 이들이 이벤트를 감지하면 이를 주변의 이웃들에게 알려 각자의 감지영역을 체크하도록 메시지를 전송한다. 메시지를 받은 센서들은 자신의 감지영역을 체크하고 어떤 조건들을 만족하는지 여부에 따라 깨어날 것인지 계속 sleeping 모드에 있을 것인지를 결정한다. 각 센서에서 실행되는 이 알고리즘은 매우 간단하며 필요에 따라 기껏해야 2 bit의 메시지를 전송하게 된다. 제안된 알고리즘에서 각 센서들은 주어진 네트워크에 의해 초기에 결정되는 하나의 정보-통신 반경 내 센서의 개수 혹은 이웃의 수-를 제외하고는, GPS 정보나 이웃들로부터의 상대적인 위치정보나 거리 등과 같은 정보들은 전혀 이용하지 않는다.

이 알고리즘의 또 하나의 장점은 위에서 설명한 “wake-up 모드가 되기 위한 어떤 조건”에 사용되는 두 가지 역치값들의 적용에 있는데, 이 역치값들을 조절함에 의해서, 우리는 이벤트 감지의 정확도와 통신 비용간의 trade-off를 조종할 수 있기 때문이다.

키워드 : 센서 네트워크, 분산 알고리즘, 이벤트 감지

Abstract In this paper, we present a distributed algorithm for detecting events in wireless sensor network that provides trade-off between detection accuracy and costs. Our model is that the region of interest is monitored by a large number of randomly distributed 0/1 sensors with low-power battery, limited functionality and memory, just enough for sensing, communicating and performing simple arithmetic operations.

For any event detection problem, one needs some sensors awake to detect the event at the time that it happens, and to wake up its neighbors to detect further events. Scheduling for the network to save the total power-cost or to maximize the monitoring time has been studied extensively. Our scheme is that some predetermined uniformly distributed sensors, called critical sensors, are awake all

· 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2010-0008959)

† 종신회원 : 숭실대학교 컴퓨터학부 교수
hsnaa@ssu.ac.kr
†† 학생회원 : 숭실대학교 컴퓨터학부
dkhtanh@gmail.com

논문접수 : 2010년 10월 11일
심사완료 : 2011년 1월 10일

정보과학회논문지: 정보통신 제38권 제2호(2011.4)

the time and when a critical sensor detect any event, it broadcasts to the neighbors to check their sensing area. Then the neighbors check their area and decide with certain criteria whether they wake up or remain in sleeping mode . This algorithm running in each sensor is very simple and uses at most 2 bit of broadcasting. Sensors do not need any information such as GPS data or the relative position from the neighbors, but the number of its communication neighbors which is initially determined when the network is set.

Another advantage of our algorithm is that we adapt two kinds of measure for the wake-up decision. By adjusting the threshold values, our algorithm can be applied for many applications because these thresholds provide trade-off between the accuracy of event detection and the cost of energy and communication.

Key words : Sensor Network, Distributed Algorithm, Event Detection

1. Introduction

Wireless sensor network (WSN) becomes an important area of many real world applications and research activities, e.g. continuous monitoring of environment or animals or event detection [1]. It offers the ability to monitor large scale areas in which wired networks maybe unavailable or infeasible to establish. The essential requirement for WSN is that the union of sensing area of all sensors covers the region of interest. In monitoring applications, sensor nodes regularly report sensory readings. Event detection applications, however, are concerned with detecting events and sensor nodes report data only when an event is detected. In this paper, we focus on event detection only in WSN. Many applications, such as region surveillance, gas leaking detection, pollution detecting and radiation prevention [2], concern only event detection.

An important issue in sensor network using sensors with low-power battery is energy saving. Scheduling for WSN to save the total power-cost or to maximize the monitoring time has been studied extensively. Among the approaches, we are only interested in detecting event region with the least cost of awaking sensors. We only need some to be awake to detect the coming event while the others could reduce their function to recharge or save energy. But, inactivating sensors gives rise of less event detecting. There cannot be any perfect solution to achieve both the energy-saving and the accuracy of event detection in this case. Many researchers [3-6] have studied about event region detection in WSN, focusing only on one of these two aspects. In this paper we will provide a way to properly trade-off between them according to the situations.

In [6], the authors focused on energy-saving of the whole network. Each sensor node sleeps most of the time and wakes up for T0 in every T time unit. While in active mode, sensor would detect if there is any event around. Their approach is good if we can choose a good time period, yet all sensors need to periodically wake up even if there is no event. In [4], the authors studied a good fault-tolerant distributed way of detecting event. Intuition behind their work is that the difference of measuring values between event sensors and normal sensors is big. Sensors whose measuring values differ a lot from those of the neighbors are defined as boundary sensors, and such boundary sensor could be locally detected by simply comparing its value with those of neighbors. Yet, in energy-efficiency, they require all the network to wake up all the time to keep track on the events. Zhu et al. [5] studied on creating a contour boundary of event regions which could dynamically changing with the events. Building this contour is a good way to study the topology of events, but this contour is easily broken if the bandwidth of the boundary is thin. Moreover, as in [4], in order to keep track of the event changing, they need all sensors to wake all the time. Also, in our setting, most of sensors are sleeping, so when an event is detected, broader region than the boundary contour needs awake in order to detect most of nearby events.

A most similar approach to ours is the work in [3]. The authors used two types of sensors, tripwire sensors, which are inexpensive, low-battery sensors with limited processing and communication capabilities, and tracker sensors, which are specialized sensors with high capabilities of processing and communication. Since the energy cost of tracker sensors is high, most of the time they will sleep to conserve energy; on the other hand, tripwire sensors with low power are awake all the time, to catch events and wake up tracker sensors. When an event is detected by a tripwire sensor, they wake up some (not every) appropriate tracker sensors in the neighborhood of the tripwire sensor. Their purpose is to save energy of tracker sensors.

Our scheme is that some predetermined sensors, called critical sensors, are awake all the time and when the event is detected by a critical sensor the sensor broadcasts to the neighbors to check their sensing area. Then the neighbors check their area and decide with certain criteria whether they wake up or remain in sleeping mode. For such wake-up decision, each sensor calculates how many neighbors are seeing the event now and how many neighbors are awake. Both are the indicators for how close the event is from the sensor, but the former is more for detecting the current detected events' activity while the latter for waiting events that are likely to appear.

Our approach differs from that of [3] in three aspects: Firstly, our sensors are all identical, inexpensive, low-battery sensors with limited processing and communication capabilities. Secondly, we randomly choose some sensors as the 'guards' waking up all the time, called critical sensors, with approximately small constant density, while the others are sleeping before certain criteria are satisfied. Lastly, our purpose is to save the total energy of sensors (which will be proportional to the number of wake sensors). Thus, simulating the approach of [3] in our network model and comparing the results is meaningless. In Section 4, we compare all the simulation results of our approach with the counterpart of traditional approach such that all sleeping sensors become awake right after they got any message from the neighbors.

The rest of the paper is organized as follows: In Section 2, we describe our model of the network and the distributed algorithm. In Section 3, we explain for how to adapt our algorithm for many applications by adjusting the threshold values in the wake-up criteria. In Section 4, we present our simulation results.

2. Distributed algorithm to detect events

2.1 The model of the network

Our sensors are identical 0-1 sensors. We can either say that each sensor contains a certain predicate that outputs 1 if sensor reading is within some specific range of interest to us and 0 otherwise. Each sensor has low-powered battery and limited memory just enough to perform simple arithmetic operations. The sensing (resp. communication) area of one sensor is a disk of radius r_s (resp. r_c). Each sensor has two modes: SLEEP or WAKE. In SLEEP mode, the only functionality that a sensor performs is receiving messages and only when asked, checking its sensing area or performing simple arithmetic operations, in order to decide whether or not it must switch to WAKE mode. In WAKE mode, a sensor monitors its sensing area all the time and broadcasts some messages whenever needed.

Each sensor s_i maintains the following information:

- The mode is SLEEP or WAKE
- The event value $v(i)$: 1 if its reading is within the range of interest, and 0 otherwise. If $v(i)=1$, we call S_i event sensor.
- The event ratio $er(i)$: the ratio of the number of neighboring event sensors over the total number of neighbors.
- The wake ratio $wr(i)$: the ratio of the number of awake neighbors over the total number of neighbors.

We partition the region of interest into many sub-regions, each covered by randomly distributed disks of radius r_s centered at point-sensors with a given density (see Fig. 1). Then, by a natural assumption $r_s \le r_c$ it is guaranteed that the communication network generated by all deployed sensors is connected, so that some base station(s) can collect information on all detected events. However, in order to simplify the visualization, in the figures below, we assume that $r_s = r_c$.

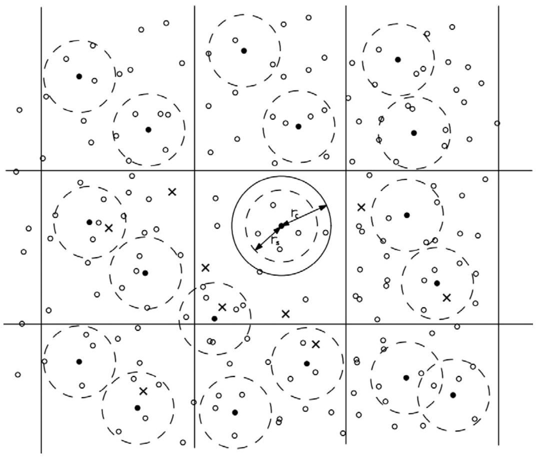


Fig. 1 Illustration of our model of the network. The region of interest is divided into squares, each monitored by randomly distributed sensors (tiny circles). Black tiny discs indicate critical sensors that are always in fully functional state. Their sensing area of radius r_s and communication area of radius r_c are drawn in dashed and solid line; here only critical sensor's communication area is shown. Events are depicted by cross marks, so the five cross marks within the dashed circles indicate the events detected by critical sensors at the moment that they appear.

Let $S=\{s_1,...,s_N\}$ be the set of all deployed sensors in the region of our interest. We randomly choose fixed number of sensors in each sub-region, called critical sensors, and denote their set by S_c. To save the total energy-consumption, we make only the critical sensors monitoring their regions all the time, while the others in $S\setminus S_c$ are in SLEEP mode. The way to choose critical sensors is itself an important research topic in network society and is not an issue of our research. So, we just assume that critical sensors are carefully chosen so as to satisfy the coverage condition of the sensor network: The whole region of interest is always monitored by critical sensors.

We may cycle the monitoring role after grouping the critical sensors. Scheduling for the network to save the total power-cost or to maximize the monitoring time has been studied extensively. We assume that each sensor has knowledge of the number of its neighbors within the communication network, but neither of its global location nor of the relative position or distance from its neighbors.

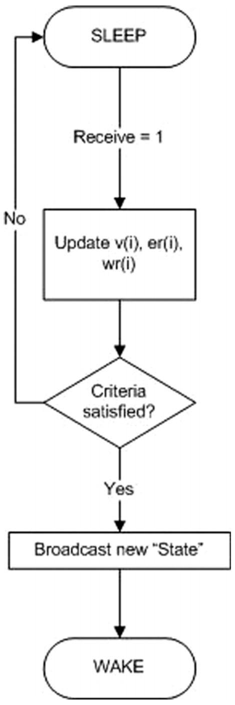


Fig. 2 The algorithm run by sensor s_i

2.2 The algorithm

Our goal is to detect events in a distributed way (i.e., without using any global computation or knowledge of geometric information of sensors and their neighbors), and in an energy-efficient way where only critical sensors are in WAKE mode all the time and the others stay sleeping unless the signals from the neighbors are cumulated enough to decide that the event is nearby. We encapsulate such intention to define the following criteria for WAKE mode:

[Criteria] Let $0 \le \alpha, \beta \le 1$ be the prescribed thresholds. Sensor s_i moves to WAKE mode if one of the following conditions are satisfied and otherwise to SLEEP mode: (1) Event value $v(i)$ is 1; (2) Event ratio $er(i) > \alpha$; (3) Wake ratio $wr(i) > \beta$.

The first condition implies that some event is being detected in the sensing area of this sensor, so it needs to move to WAKE mode. The second one tells us that more than α portion of the neighbors are detecting event now, so even if there is no event detected around this sensor, it needs awake since the event is nearby. The last condition means that more than β portion of the neighbors have been awake, so with high probability the event is coming even if events have not been detected yet in the neighborhood.

Now our simple distributed algorithm designed to run in a sensor is shown in Fig. 2: if a sensor receives a message from a neighbor or it detects event in WAKE mode, the sensor checks first the above criteria. Depending on the result of checking, the sensor moves to SLEEP or WAKE mode and broadcast its new state to the neighbors once if necessary. Broadcasting a message is necessary when the mode is changed or the event value $v(i)$ is changed. For instance, if a sensor, previously not detecting any event but awake by the second condition ($er(i) \geq \alpha$), detects an event, then the mode is unchanged, but the sensor needs to broadcast the change of event value $v(i)$ from 0 to 1 because it will change event ratio $er()$ of its neighbors and possibly some of them become awake. All change of a sensor can be simply encoded into 2 bits and thus broadcasting or receiving a message costs very little computation. Also, as shown in Fig. 2, a sensor broadcasts a message at most once for one check of criteria. The algorithm runs in distributed way by individual sensors without any global information or relative location from the neighbors.

Fig. 3 illustrates the results of our distributed algorithm step by step, where $\alpha = 0.3$ and $\beta = 0.48$ are prescribed in all sensors. Sensors are depicted by tiny circles and the events by crosses. The first figure illustrates the beginning of the event detection. As soon as the two events happen, among the critical sensors monitoring the area all the time, only one critical sensor (whose monitoring area is indicated by the big circle) is within the distance r_s from the events and thus detects an event. The critical sensor broadcasts its detection of an event (its event value becomes one), which is followed by its seven neighbors (depicted by black disks in the second figure) becoming event sensors. The situation converges to the third figure, unless there is no change of events; around the first detected event,

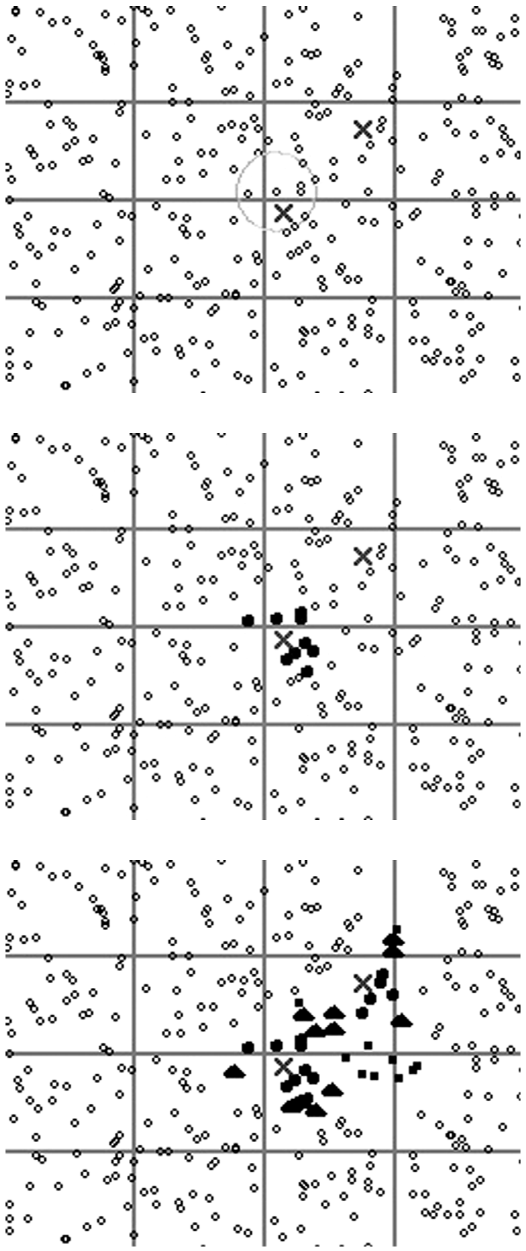


Fig. 3 Illustration of the algorithm when $\alpha = 0.3$ and $\beta = 0.48$; Sensors are depicted by tiny circles, events by crosses, event sensors by black disks, sensors in WAKE mode either by triangles (if its event ratio is greater than α) or by squares (if its wake ratio is greater than β).

some sensors (depicted by black triangles) are converted to WAKE mode because their event ratio $er()$ become greater than α. Also, some other sensors (depicted by black squares) become awake because their wake ratio $wr()$ become greater than β. Finally, these sensors wake up the sensors nearby the undetected event (the upper-right cross mark), which yields the detection of a new event.

2.3 Analysis of the communication cost

For a given network, when events happen, for all sensors s_j, the values $v(j), er(j)$, and $wr(j)$ increase only and thus mode-change happens only from SLEEP to WAKE; no awake sensor in the network moves to sleep. So, a sensor changes the mode at most once and thus broadcasts a message encoded in 2 bits at most once (as shown in Fig. 2). Therefore, the communication cost of updating the status of the network incurred by the events is at most two times the number of sensors in WAKE mode. In particular, the communication cost by the events is at most two times the size of the network.

3. Control of thresholds α and β

Our algorithm wakes up not only the event sensors but also those nearby the events. In this section, we discuss how we use two thresholds α and β to wake up sensors that are reasonably near to the events but in distance more than r_s from any event (thus not event sensors), without measuring the distance and increasing the communication cost much. Thus, our algorithm can be applied in many real applications by choosing thresholds α, β suitable to their given network.

3.1 Threshold α

Event ratio $er(i)$ is the ratio of event detecting neighbors over all neighbors of s_i, so the higher this is, the more likely the event is not yet detected but will come shortly to the position of s_i.

Denote the communication region of s_i (disc of radius r_c centered at s_i) by $B(s_i,r_c)$ and the disc of radius r_s centered at q by $B(q,r_s)$. Assuming that both of sensors and events are distributed with reasonable density, we know that the expected value of $er(i)$ is the ratio of area of $\bigcup_{event\, q} B(q,r_s) \cap B(s_i,r_c)$ with area of $B(s_i,r_c)$ for all sensors s_i. Thus, if sensor s_i is at distance more than $r_c + r_s$ from any event, $er(i)$ is expected to be zero, so for more interesting situations, let the sensor s_i

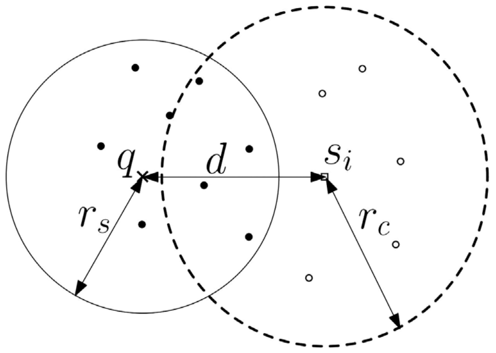


Fig. 4 Illustration of how α works

lie within distance $0 < d < r_c + r_s$ from some event q. Then $B(s_i,r_c)$ intersects the disc $B(q,r_c)$ as shown in Fig. 4, and the ratio of area of $B(q,r_s) \cap B(s_i,r_c)$ with area of $B(s_i,r_c)$ can be expressed as the following and the expected value of $er(i)$ is at least this:

$$\text{if } r_c - r_s < d < r_c + r_s,$$
$$\begin{cases} S = -\left(\frac{1}{r_c}\right)^2 \sqrt{(d+r_c+r_s)(d+r_c-r_s)(d-r_c+r_s)(-d+r_c+r_s)} \\ er(i) = \frac{1}{\pi}\left\{\cos^{-1}\frac{r_c^2+d^2-r_s^2}{2r_c d} + \left(\frac{r_s}{r_c}\right)^2 \cos^{-1}\frac{r_s^2+d^2-r_c^2}{2r_s d} + S\right\} \end{cases}$$
$$otherwise\,(0 < d \le r_c - r_s),\ er(i) = \frac{\pi r_s^2}{\pi r_c^2} = \left(\frac{r_s}{r_c}\right)^2.$$

To get some intuition of this value, we consider a simple case where $r_s = r_c$. We start with the traditional approach in sensor network that wakes up sensors within distance $d = r_s$ from any event (that is, to wake up only event sensors). Then we plug $d = r_s = r_c$ into the formula and obtain that the ratio is $\frac{2}{3} - \frac{\sqrt{3}}{2} \approx 0.39$.

This means that with reasonable density of sensors and events, $B(s_i,r_c)$ is likely to intersect at least one of discs $B(q,r_s)$ and thus $er(i)$ is expected to be greater than 0.39. So, even with setting $\alpha = 0.39$, our algorithm can speed up the event detection because the algorithm is likely to wake up sensors that are in distance more than r_s from any event and thus not currently detecting any event. In conclusion, this technique enables us to detect the events efficiently in distributed way, without increasing the communication cost too much. Indeed, even if we set $\alpha = 0$, non-event

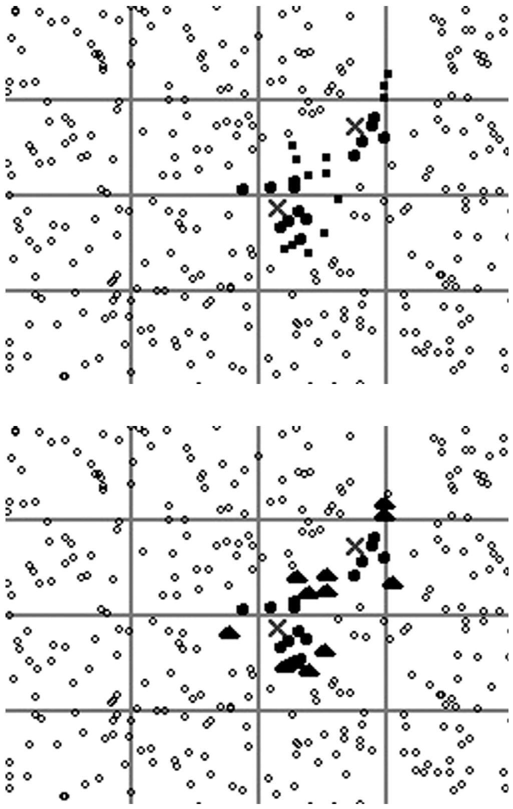


Fig. 5 Illustration of applying only $\alpha = 0.23$ by setting $\beta = 1$, or only $\beta = 0.44$ by setting $\alpha = 1$

awake sensors are not ranged unlimited far from the events. The left image of Fig. 5 is obtained by running our algorithm in the same network and events with $\alpha = 0.23$ and $\beta = 1$. With $\beta = 1$ fixed, the event depicted by the top-right cross has not been detected before reducing α to 0.23.

Another usage of this threshold is that instead of setting β too low (which ends up with too many awake sensors and high costs), we can apply lower α in order to wake up most of the sensors around the events, and higher β to avoid high costs.

3.2 Threshold β

Wake ratio $wr(i)$ is the ratio of awake neighbors among all neighbors, so the higher this is, the more likely the events are not detected yet but nearby and the sensors better to be awake for detection.

Controling the threshold β provides a trade-off between the efficiency of detection and the communication cost; If $\beta = 0$, then all the sensors in S become awake, and if $\beta = 1$, then only event sensors or sensors with $er() \geq \alpha$ become awake. If β is too small, then the number of awake sensors and the communication cost become too high, thus the network becomes slow by overload and cannot easily cope with the event change. On the other hand, if β is too big, then the detection spreads only by satisfying that $er(i) \geq \alpha$ and the network may miss many events. Thus, our suggestion is to use relatively low β but not to allow the waking-up process beyond a reasonable number of hops from the event sensors. The right image of Fig. 5 is obtained by running our algorithm in the same network and events with $\alpha = 1$ and $\beta = 0.44$. With $\alpha = 1$ fixed, the event depicted by the top-right cross has not been detected before reducing β to 0.44.

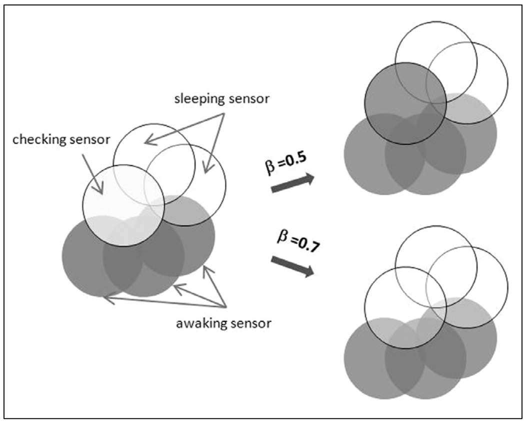


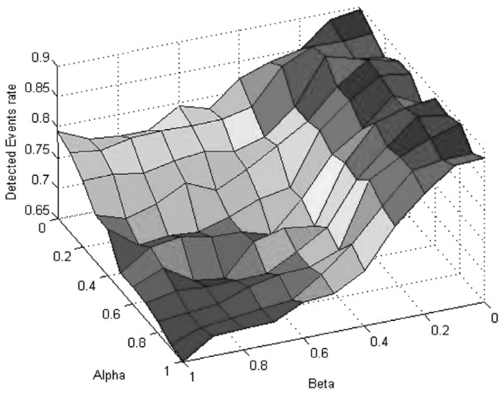
Fig. 6 Illustration of how β works

4. Simulation results

In this section, we report our simulation results. We implemented our distributed event detection algorithm in C++. Every network we generated in the simulations consists of more than 2500 sensors that are independently and identically distributed from the uniform distribution in a field of size 1024 × 768 units. In this field, we partitioned the region into subregion the side of 128×48 units. In each subregion we randomly distributed 20 sensors, and among them we randomly choose 2 sensors as critical sensors. For simplicity of demonstration, each sensor is set to have the same communication radius and sensing radius as 20 units. We are not interested in dependency of the communication cost on the density of sensors, since many previous works have already proved that with the increase of the density of sensors, the rate of events being detected is getting higher.

4.1 Trade-off between the detection accuracy and costs using thresholds α, β

Fig. 7 illustrates that the lower the thresholds are, the higher the rate of event detection and the rate of WAKE sensors are. It also shows that determining α and β is a matter of balancing the trade-off between accuracy of event detection and the communication and energy cost. For each pair (α,β), we repeat the following more than 100 times and plot the average rates: (i) we generate a random network (described above) and 200 random events, and (ii) we run our algorithm and compute the rate of event detection and the rate of WAKE sensors.



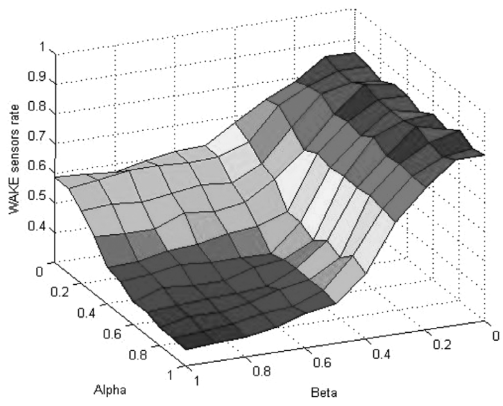


Fig. 7 Dependency of the rate of event detection and of WAKE sensors, on thresholds α, β

4.2 Comparison of the performance between our algorithm and the traditional method

In this subsection, we compare the performance of our approach with that of the most popular and easy-to-implement method (which we call "traditional method") that wakes up as many sensors as possible in all direction.

Table 1 illustrates the result of our experiment: We fix a specific sample of randomly distributed sensors and events from uniform distribution over the field, and run our algorithm with three pairs of thresholds (α,β), being (0.2,0.2), (0.5,0.5), (0.9,0.9), allowing only k-hops of communication between sensors. We compare these results with the traditional approach, in both detected event rate and wake sensors rate. For the analysis, consider the case where k=3. We observe that the detected events rate is 0.78 in traditional approach, while with thresholds (0.2,0.2), our approach achieves only 0.77. However, with sacrificing this 1% of loss in detection accuracy, our approach gains 4.6% of saving in the energy or communication cost in comparison of the wake sensors rate to that of traditional approach. This phenomenon becomes more dramatic with the other pairs of thresholds; with (0.5,0.5), 6.4% of loss in detection accuracy achieves almost 22% of saving in cost, and with (0.9,0.9), 7.2% of loss in detection accuracy gives almost 27% of saving in cost, when compared with the traditional approach. Thus our algorithm is very practical in the sense of providing trade-off between detection accuracy and costs, and is very effective for energy-saving with small loss of detection accuracy.

For more visual illustration of Table 1, we adapt a curve whose x-axis represents the detected events rate, and whose y-axis denotes the wake sensors rate. This curve outlines the relationship between detection-accuracy and costs for varying threshold values of α or β. One interesting property of this curve is that the one with lower slope is the one which could create a boost in detection rate while reducing the cost of energy. From Fig. 8, we observe that the higher α and β are, the more efficient in gaining the detection accuracy. Note also that every method has limitation of detection accuracy; the limit of detection accuracy

Table 1 Trade-off between Detected Event Rate (DER) and Wake Sensors Rate (WSR) when only k-hops of communication is allowed

k	1		2		3		4		5	
	DER	WSR	DER	WSR	DER	WSR	DER	WSR	DER	WSR
Traditional approach	0.374	0.056	0.708	0.522	0.780	0.657	0.816	0.728	0.854	0.789
$\alpha=0.2$ $\beta=0.2$	0.374	0.056	0.708	0.500	0.770	0.611	0.798	0.666	0.818	0.709
$\alpha=0.5$ $\beta=0.5$	0.374	0.056	0.708	0.419	0.716	0.439	0.716	0.444	0.726	0.454
$\alpha=0.9$ $\beta=0.9$	0.374	0.056	0.708	0.392	0.708	0.392	0.708	0.392	0.708	0.392

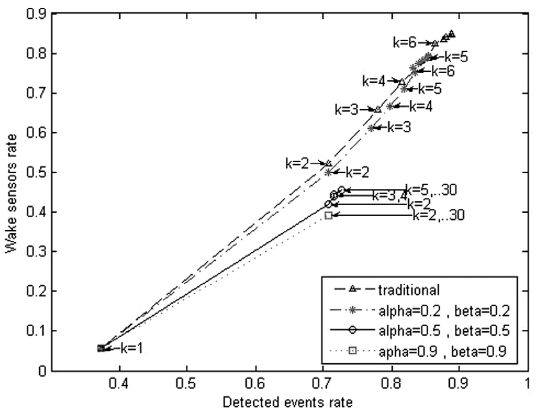


Fig. 8 Comparison with traditional approach

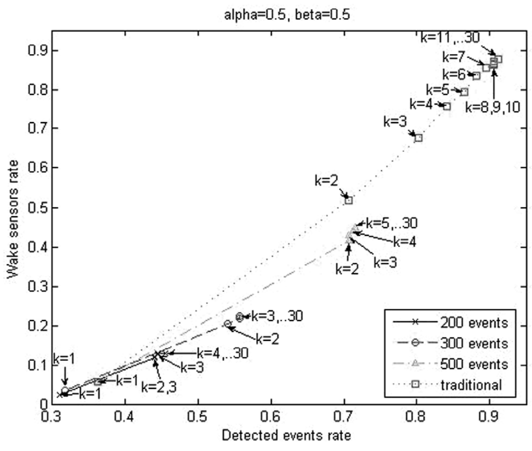


Fig. 9 Event density affection

is 70.8% with thresholds (0.9,0.9), 72.6% with (0.5,0.5), 85.4% with (0.2,0.2) and 88.8% with traditional approach.

Before concluding this paper, we present one more result. Fig. 9 is generated by executing our algorithm with thresholds (α,β)=(0.5,0.5) for a fixed network and three different sets of randomly distributed events, consisting of 200 events, 300 events and 500 events; The traditional curve is generated from a set of 500 events. From the figure, we see that the more the events occur, the more effective our approach becomes.

5. Conclusion and future works

In this paper we study the problem of event detecting with wireless sensor and propose an approach that is distributed and most likely energy-saving. We also provide 2 thresholds and a limit of k-hops-communication in order to be applied in a more flexible situation with balancing the trade-off between energy saving and effectiveness of events detection.

By using the two thresholds α and β, we proposed a new approach to event detecting task in an assumption that the system is working under the condition of energy saving. Instead of blindly extend the monitoring region as in traditional methods, we use only local judgement, based on neighboring situation. In the aspect of energy saving, our approach performs best, achieving reasonable accuracy in detection. With a simple algorithm running in sensors, our method only requires local information (the number of 1-hop neighbors), and does not need any type of global data, so this should be adaptable in distributed system. In the aspect of communication cost, each message sent by sensors needs only 2 bits and does not require any reply.

We believe that our method can be applied to dynamic events, which would be one of future works. We also plan to take into account the failure-tolerance property of wireless sensor network. Also, despite the fact that the two thresholds provide the flexibility, we need to study about how to optimize them to improve the performance.

References

[1] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," *Proceedings of the 4th international symposium on Information processing in sensor networks*, pp.497-502, 2005.

[2] S. Brennan, A. Mielke, and D. Torney, "Radioactive Source Detection by Sensor Networks," *IEEE Transactions on Nuclear Science*, vol.52, pp. 2273-2278, 2005.

[3] Thrasyvoulos Spyropoulos, C. S. Raghavendra, Viktor Prasanna, "A Distributed Algorithm for Waking-up in Heterogeneous Sensor Networks," *Springer-Verlag Lecture Notes in Computer Science (LNCS) 2634 (from IPSN'03)*, Apr. 2003.

[4] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, P. Deng, "Localized Outlying and Boundary Data Detection in Sensor Networks," *IEEE Transactions on Knowledge and Data Engineering*, pp.1145-1157, 2007.

[5] X. Zhu, R. Sarkar, J. Gao, J. Mitchell, "Light-weight Contour Tracking in Wireless Sensor Networks," *IEEE INFOCOM 2008*, pp.1175-1183, 2008.

[6] Y. Zhu, Y. Liu, L. M. Ni, and Z. Zhang, "Low-Power Distributed Event Detection in Wireless Sensor Networks," *IEEE INFOCOM 2007*, pp.2401-2405, 2007.

나 현 숙

1993년 서울대학교 수학과(학사). 1995년 포항공과대학교 수학과(석사). 2002년 포항공과대학교 수학과(박사). 2001년 3월~2002월 8월 프랑스 INRIA Post Doc. 2002년 9월~2003년 2월 홍콩과학기술대(HKUST) 전산학과 Post Doc. 2003년 3월~현재 숭실대학교 컴퓨터학부(부교수). 관심분야는 알고리즘, 계산기하학

Nhu Tuan Anh

2007년 IT Deparment, Natural Science University, Hochimihn city, Vietnam (학사). 2010년 8월 숭실대학교 컴퓨터학부(석사). 2010년 9월~현재 숭실대학교 컴퓨터학부 박사과정