Distributed Event Detection in Wireless Sensor Networks for Disaster Management

Majid Bahrepour, Nirvana Meratnia, Mannes Poel, Zahra Taghikhaki, Paul J.M. Havinga
University of Twente
7500 AE, Enschede, Netherlands
(m.bahrepour, n.meratnia, mpoel, z.taghikhaki, p.j.m.havinga) @utwente.nl

ABSTRACT -Recently, wireless sensor networks (WSNs) have become mature enough to go beyond being simple fine-grained continuous monitoring platforms and become one of the enabling technologies for disaster early-warning systems. Event detection functionality of WSNs can be of great help and importance for (near) real-time detection of, for example, meteorological natural hazards and wild and residential fires. From the data-mining perspective, many real world events exhibit specific patterns, which can be detected by applying machine learning (ML) techniques. In this paper, we introduce ML techniques for distributed event detection in WSNs and evaluate their performance and applicability for early detection of disasters, specifically residential fires. To this end, we present a distributed event detection approach incorporating a novel reputation-based voting and the decision tree and evaluate its performance in terms of detection accuracy and time complexity.

Keywords -Disaster early warning systems, event detection, wireless sensor networks

I. INTRODUCTION

Disaster management or emergency management is a key discipline for providing necessary responses whenever and wherever a catastrophe occurs to save lives and reduce casualties. From an engineering approach, machineries can be designed and used to help with detection or prediction of the disastrous events. One of the recent technologies enabling (near) real-time detection of such events is the wireless sensor networks (WSNs). Wireless sensor networks typically consist of a large number of small, low-cost sensor nodes distributed over a large area. The sensor nodes are integrated with sensing, processing and wireless communication capabilities. Each node is usually equipped with a wireless radio transceiver, a small microcontroller, a power source, and multi-type sensors (e.g. temperature, humidity, smoke). These components enable a sensor node to sense the environment, communicate and exchange sensory data with other nodes in the area, locally process its own data and make smart decisions about what it observes. This will lead to detection of events and unusual data behaviors whenever and wherever they occur. This feature is called *event detection*. Event detection functionality of WSNs has attracted much attention in variety of applications such as industrial safety and security, meteorological hazards, and fire detection [1].

Resource constraints of the wireless sensor nodes, dynamicity of the deployment area [2], and unreliability of wireless communication introduce unique design challenges. As a result, event detection techniques for WSNs need to be light-weight (to meet limited computational capability of the sensor nodes), distributed (to split big processes into several smaller segments to facilitate parallel processing), and robust against sensor node and wireless communication failures. It must be accurate to reduce number of false alarms and to prevent creating unnecessary chaos and stress. In addition, it needs to detect disastrous events fast, as this is the first step towards creating awareness and generating timely alarms.

Developing an event detection approach meeting all the aforementioned requirements is not a trivial task and many of the existing approaches often only partially meet these requirements.

In this paper, we propose a light-weight and accurate event detection approach for in-network decentralized event detection. The proposed approach uses decision trees for distributed event detection and a novel reputation-based voting method for aggregating the detection results of individual sensor nodes and reaching a consensus among different decisions. We will show that despite their simplicity, decision trees are highly accurate and their simplicity fulfills the WSNs requirements. The performance of the proposed approach is gauged in terms of detection accuracy and time complexity.

II. RELATED WORK

The problem of event detection has been tackled from different perspectives. The basic idea of event detection can be defining some threshold values then, an alarm is generated when sensor readings are lower or higher than a pre-defined threshold value [3-6]. Due to the fact that events are often sophisticated and cannot be detected just by simple pre-defined thresholds, the new trend in event detection is to use pattern matching or machine learning techniques. Based on scale of the network, application requirements and constraints, pattern matching have been proposed to be conducted in the base station [7, 8], locally in the sensor nodes [9], or distributed over the network [10-14]. Among these three, distributed pattern matching fulfills best the requirements of WSNs. This is because involvement of more than one sensor node makes the whole event detection process more robust against sensor failures. Additionally, since the detection is done in the network, communicational overhead to the base station is reduced and energy consumption is decreased.

Naïve Bayes [9], feed forward neural networks [9], and support vector machines (SVM) [15] have been proposed to detect an event locally on single individual sensor nodes.

Distributed in-network studies incorporating collaboration between nodes and data exchanges, however, range from techniques based on distributed fuzzy engine [16], map base pattern matching [17], feed forward neural networks, and naïve Bayes classifiers [18].

III. DECISION-TREE BASED EVENT DETECTION

To fulfill WSNs requirements mentioned previously, our proposed approach is a distributed machine learning (ML) technique that uses decision trees to detect events in a distributed manner. Unlike many other complicate approaches, we will show that simplicity of the decision tress is what WSNs exactly need and our approach can fulfill both low computational overhead and high detection accuracy.

A decision tree is a learning algorithm that uses tree-like graphs to model and evaluate discrete functions [19, 20]. The inputs to the tree might be either continuous or discrete but the outputs (the decisions) are always discrete. Construction of a decision tree for classification requires a training phase. This training phase employs a set of data and a learning algorithm to find a minimum depth decision tree. The tree should contain the minimum required nodes (or minimum depth) to reduce time and memory complexities. Therefore, the training algorithm is usually a local search greedy algorithm to find an optimum decision tree.

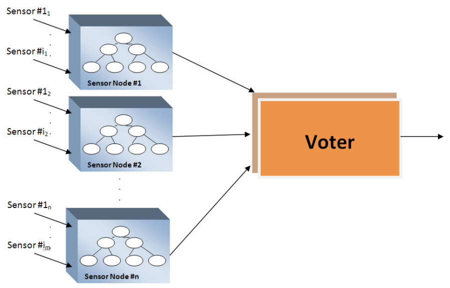

Figure 1: Block diagram of the proposed approach.

As it can be seen from Figure 1, in our proposed approach every individual sensor node performs event detection using its own decision tree-based classifier and the classification results, i.e., detected events, from various nodes are aggregated by a voter running a reputation-based voting technique. To show superiority of our reputation-based voting, we additionally investigate three other voting mechanisms based on the classical majority voting.

A. *Reputation-based Voting*

Once each node makes its individual decision about occurrence of an event, a consensus needs to be reached. One of the mechanisms to reach this consensus is through use of voting. There are various voting techniques, among which is the reputation-based. Reputation-based voting approaches are based on finding reputation of individual sensor nodes and choosing the decision made by the nodes having the highest reputation. To use the reputation-based technique in our event detection technique, sensor nodes must first run their local decision tree classifiers. Then assuming that sensor nodes have detected events correctly, they should judge how well the other sensor nodes could detect events. To do the judgment, each sensor node first sends its detected event, called Detection Value (DV), to all other nodes in its neighborhood. The DVs received from the neighbors will be stored in a table called Neighbors Detection Value Table (NDVT). In the next step, each sensor node should judge about its neighboring sensor nodes by considering itself as the reference. The judgment is accomplished by comparing the difference between value of sensor node itself and value of the other sensor nodes. If the difference is less than a threshold value θ (which is chosen based on the context), the judging sensor node gives a positive vote ($V^{new} = V^{old} + 1$) to the other sensor node. Otherwise, the "being judged" sensor node receives a negative vote ($V^{new} = V^{old} - 1$). Finally, NDVT tables are sent to the voter (e.g., a cluster head) to reach a consensus among different opinions. The challenging part of reputation-based voting is how to assign a global reputation value to each sensor node in order to choose high reputed node and its detected event as the result of event detection. In what follows, we introduce two reputation-based voting techniques to assign a global reputation value to each sensor node and to reach a consensus about the detected event.

1) *Reputation Technique 1*

The reputation technique 1 checks local reputation of every individual sensor node for event detection from the other sensor nodes' perspective. The local reputation value is obtained based on average value of V^i (positive or negative votes which were given by the other sensor nodes) for each sensor node. Then, the average local reputation is multiplied by the weight of sensor nodes calculated using Eq. 1 to assign global reputation values. The event with the highest reputation weight (W) is the result of the voting procedure. Eq. 1 shows how the weights are calculated.

$$W_i = R_i \times Acc_i \qquad \text{Eq. (1)}$$

Where, W_i is the reputation value corresponding to sensor node i, R_i is the local reputation value of sensor node i from other sensor nodes' perspective, and Acc_i is the same output of Eq. 6 (weight of sensor node$_i$ for event type$_q$).

2) *Reputation Technique 2*

In reputation technique 2, we define two threshold values called θ_1, θ_2. Comparing the local reputation value (R_i) with θ_1 and θ_2 gives an insight about how well the sensor nodes detect events. If ($R_i \geq \theta_1$), then sensor nodes make perfect decisions, if $(\theta_1 > R_i \geq \theta_2)$ then sensor node make OK decisions, and if $(\theta_2 \geq R_i)$ then sensor nodes make poor decisions. Then we assign a discrete value for different performances. To do so, 0.5 indicates poor performance, 1 indicates normal performance, and 2 indicates very good performance. Reputation technique 2 is performed using Eq. 2 based on the performance of each sensor node.

$$W_i = S_i \times Acc_i \qquad \text{Eq. (2)}$$

Where, W_i is the reputation value corresponding to sensor node *i*, S_i is obtained from Eq. 3, and Acc_i the same output of Eq. 6 (weight of sensor node$_i$ for event type$_q$).

$$S_i = \begin{cases} 2 & if\ (R_i \geq \theta_1) \\ 1 & if\ (\theta_1 > R_i \geq \theta_2) \\ 0.5 & if\ (\theta_2 \geq R_i) \end{cases} \qquad \text{Eq. (3)}$$

θ_1 and θ_2 are application dependant and are chosen by developers.

To have a means for comparing the reputation-based voting, in the following subsections three other voting techniques based on the classical majority voting are presented. In Section V, a number of experiments are conducted and the results are compared.

B. Majority Voting 1: Sensor type-based Weighting

In this majority voting technique, we use contribution of each sensor to the event detection process as weight. Sensor contribution will be calculated by running the same classifier used for event detection on individual sensor nodes. To this end, each sensor node receives a weight calculated by Eq. 4 depending on the availability of sensor types.

$$Weigh\ of\ Sensor\ Node_i = \sum_{K=1}^{m} Weight\ of\ Sensor\ Type_K \qquad \text{Eq.(4)}$$

Where 'm' is the total number of sensor types in a (heterogeneous) network and 'Weight of Sensor Type $_k$' is contribution of sensor type$_K$ to event detection that is calculated by Eq. 5.

$$Weigh\ of\ Sensor\ Type_k = \frac{Detection\ Accuracy\ Using\ only\ Sensor\ Type_k}{\sum_{p=1}^{m} Detection\ Accuracy\ Using\ only\ Sensor\ Type_p} \qquad \text{Eq.(5)}$$

C. Majority Voting 2 – Event type-based Weighting

In majority voting 2, instead of assigning one weight to each sensor node, we assign *p* weights to each sensor node, where *p* is the number of event types. The reason of assigning more than one weight to each sensor node is to have more precise weights in order to perform the voting more accurately. For example, if a WSN detects four possible events (e.g., earthquake, fire, storm, flood) there are four weights assigned to each sensor node and according to the event detected by the node, the corresponding weight is used by the voter. Eq. 6 calculates necessary weights for each sensor node.

$$Weigh\ of\ Sensor\ Node_i\ for\ Event\ Type_q = \sum_{K=1}^{m} Weight\ of\ Sensor\ Type_K\ for\ Event\ Type_q \qquad \text{Eq.(6)}$$

Where 'm' is the total number of sensor types in a (heterogeneous) network and 'Weight of Sensor Type $_k$ for Event Type$_q$' is calculated by Eq. 7.

$$Weigh\ of\ Sensor\ Type_k\ for\ Event = Event\ Detection\ for\ Event\ Type_q = \frac{Detection\ Accuracy\ of\ Sensor\ Type_k\ for\ Event\ Type_q}{\sum_{p=1}^{m} Detection\ Accuracy\ of\ Sensor\ Type_p\ for\ Event\ Type_q} \qquad \text{Eq. (7)}$$

D. Majority Voting 3 - Event Type-based Weighting (without redundancy)

After studying majority voting techniques 1 and 2, we faced the problem of redundant weights. This means that the same event types produced by similar sensor nodes receive more weights. To prevent this problem, majority voting 3 considers all sensor nodes having the same sensor types and producing similar event types as one. By doing so, we remove redundant weights of analogous sensor nodes by unifying those sensor nodes which have the same sensors and report the same events. The rest of the voting procedure is done according to the majority voting 2. One notes that majority voting 3 is actually a pre-processing stage before doing the majority voting 2.

IV. DATA DESCRIPTION AND EXPERIMENTS

To test our approach, we consider residential fires as the disastrous event and test our approach on a residential fire dataset. In the following subsections the data and experiment methods are described.

A. Data Description

We obtain a set of residential fire data from NIST website (http://smokealarm.nist.gov/) for training and testing our approach. The training phase is conducted using 2/3 of data and testing phase is conducted on 1/3.

The obtained dataset contains flaming and smoldering fires. Additionally, some nuisance resources (e.g., data of toasting bread and lighting a cigarette that are not real fire) are added to make the detection more realistic for residential areas.

As a result, 1400 data instances were prepared in such a way that 933 instances (2/3 of whole data) were used for training and 467 instances (1/3 of whole data) for testing. The dataset contains four sensory data (features) that are temperature, ionization, photoelectric, and CO. We also perform a calibration procedure to make all the data in the same units.

B. *Experiment Method*

To test our approach on the residential fire dataset, we have to first train the decision trees then apply one of the voting techniques. Training of the decision tree is done by using 2/3 of the dataset and testing of the whole approach on the rest of the dataset. In a heterogeneous network, sensor nodes may have different sensor types. In such networks, we should either find a decision tree per each sensor node or make a decision tree that works with all sensor nodes independent of its sensor types. In this paper, we propose to make a single decision tree for all sensor types available in the dataset. Additionally, during the training phase we deliberately add some missed values per each sensor type. This is to cope with situations in which a single sensor node does not have all the sensor types. In such a case, the absent sensor types are represented by missed values.

The testing phase is conducted by feeding the same instance of data to all sensor nodes then unifying the results of event detection using one of the voting techniques. The necessary weights for voting part are obtained from contribution of each sensor, presented in Table 1, to the fire detection. It can be seen that CO is contributing the most to the fire detection process.

In the next section the results of event detection using decision trees and four aforementioned voting methods are reported.

Table 1: Contribution of each sensor to the fire detection using decision trees

	Fire Event Detection (in general)	**Flaming Fire Detection**	**Smoldering Fire Detection**	**Nuisance Detection**
Temperature	19%	26%	20%	13%
ION	16%	2%	20%	20%
Photo	23%	2%	27%	23%
CO	42%	70%	33%	44%

V. EXPERIMENTAL RESULTS

To test our event detection approach, we consider seven different network schemas presented in Table 2.

Table 2: Network Schemas

#	**Availability of sensors**					**Feature**
	Node	TMP	ION	Photo	CO	
1	1	✓	✓	×	×	Having at least one and at most two sensor types On each sensor node.
	2	×	✓	✓	×	
	3	✓	×	✓	✓	
	4	✓	×	×	✓	
	5	✓	×	×	×	
	6	×	✓	×	×	
	7	×	×	×	✓	
	Node	TMP	ION	Photo	CO	
2	1	✓	✓	×	×	Having redundant sensor types on each sensor node.
	2	×	✓	✓	×	
	3	✓	×	✓	✓	
	Node	TMP	ION	Photo	CO	
3	1	✓	×	×	×	Unavailability of CO sensor (as CO is the strongest sensor for fire detection).
	2	×	✓	×	×	
	3	×	×	✓	×	
	4	✓	✓	×	×	
	5	✓	×	✓	×	
	6	×	✓	✓	×	
	7	✓	✓	✓	×	
	Node	TMP	ION	Photo	CO	
4	1	✓	×	×	×	Having only one CO sensor (as CO is the strongest sensor for fire detection)
	2	×	✓	×	×	
	3	×	×	✓	×	
	4	✓	✓	×	×	
	5	✓	×	✓	×	
	6	×	✓	✓	×	
	7	✓	✓	✓	×	
	8	×	×	×	✓	
	Node	TMP	ION	Photo	CO	
5	1	✓	✓	×	×	Having CO sensor in less than half of the nodes
	2	×	✓	✓	×	
	3	✓	×	✓	✓	
	4	✓	×	×	✓	
	5	✓	×	×	×	
	6	×	✓	×	×	
	7	×	×	×	✓	
	Node	TMP	ION	Photo	CO	
6	1	✓	×	×	×	Having redundant sensor types on each sensor node.
	2	✓	×	×	×	
	3	✓	×	×	×	
	4	✓	×	×	×	
	5	✓	×	×	×	
	6	✓	×	×	×	
	7	✓	×	×	×	
	8	×	×	×	✓	

Table 3 reports the results of our fire event detection tests.

Table 3: Results on the Distributed Approach

Net. Architecture #	**Net. Architecture #**	**Net. Architecture #**	**Net. Architecture #**
1	Rep. technique I	96.62%	1.44
	Rep. technique II	96.57%	1.54
	V. technique #1	92.68%	8.24
	V. technique #2	94.16%	8.79
	V. technique #3	96.35%	3
2	Rep. technique I	98.18%	0.7
	Rep. technique II	93.32%	2.3
	V. technique #1	95.5%	2.92
	V. technique #2	97.43%	0.54
	V. technique #3	92.46%	6.46
3	Rep. technique I	89.64%	6.04
	Rep. technique II	79.66%	11.49
	V. technique #1	72.48%	15.37
	V. technique #2	68.25%	18.55
	V. technique #3	71.65%	17.25
4	Rep. technique I	91.39%	6.008
	Rep. technique II	84.84%	12.61
	V. technique #1	89.25%	5.82
	V. technique #2	84.37%	11.80
	V. technique #3	82.22%	9.45
5	Rep. technique I	95.78%	8.209
	Rep. technique II	94.88%	5.1478
	V. technique #1	91.52%	10.16
	V. technique #2	94.71%	7.94
	V. technique #3	95.75%	3.19
6	Rep. technique I	47.79%	11.24
	Rep. technique II	47.79%	11.24
	V. technique #1	41.92%	16.21
	V. technique #2	44.85%	19.85
	V. technique #3	93.43%	6.89

According to Table 3, we can generally say that majority voting techniques 1 and 2 have almost the same detection accuracy. And, majority voting 3, works better than majority voting 1 and 2 when there are some redundant sensor nodes.

In addition, reputation-based voting works most of the time better than the rest. However, in sixth experiment, reputation-based voting is not working well because there are redundant nodes or information in the network. One can conclude that the reputation-based voting is the best as far as there are no redundant nodes in the network.

VI. TIME COMPLEXITY ANALYSIS

Our aim is to investigate applicability of computationally intensive machine learning (ML) techniques for resource-limited wireless sensor networks. For event detection not only detection accuracy but also time complexity are important.

Time complexity of decision trees depends on two phases (1) making the decision tree (training) and (2) classification using the decision tree. Since the training part is only performed once in an offline manner, the time complexity for training phase can be ignored. In the following subsections time complexities of the approaches are investigated by only considering the time they are running in the network independent of their training part.

A. Time Complexity of the Decision Tree

The order of the decision tree appraisal is a function of the depth of decision tree and Eq. 8 presents the time complexity:

$$O(local\ approach) = O(Decision\ tree\ appraisal) \quad \text{Eq. (8)}$$

$$O(Local\ approach) = O(m) \quad \text{Eq. (9)}$$

Where m is depth of the decision tree

Once the tree is constructed by its learning algorithm, it can be pruned to reduce the number of nodes. Reducing the number of nodes helps with reducing time complexity but decrease the classification accuracy in most of circumstances, as well.

B. Time Complexity of the Proposed Approach Using Reputation Theory

Time complexity of our proposed approach using reputation theory is a function of three parts. Firstly, decision tree is evaluated (that is classification part), then local processes are performed on the nodes (local judgment), and finally consensus is reached.

$$O(Distributed\ Reputation) = \max\,[O(Decision\ tree\ appraisal) + O(process\ on\ the\ node) + O(reputation\ voting)] \quad \text{Eq. (10)}$$

$$O(Distributed\ Reputation) = Max\,[O(m) + O\left(n(n-1)\right) + O\left(\left(n(n-1)\right) + n + c\right)] \quad \text{Eq. (11)}$$

$$O(Distributed\ Reputation) = O(n^2) \quad \text{Eq. (12)}$$

Where n is the number of nodes

C. Time complexity of the Proposed Approach Using the Majority Voting 1

In the distributed approach, n sensor nodes detect events in parallel using decision trees. Therefore, the order of whole classification part is $O(m_1 + m_2 + \cdots + m_n) = O(m)$; where n is the number of nodes involved in the event detection and m is depth of the decision tree. Then these results are given to the voter to reach a consensus. Since the voting is independent from the classification, the time complexity is added to the classification time as shown in Eq. 13.

$$O(Distributed\ approach\ using\ voting\ 1) = O([m]) + O[Voting\ 1]) = Max[\,(m)\,, Voting\ 1] \quad \text{Eq. (13)}$$

$$O(Voting\ 1) = O(Assiging\ weights + Max\ finding) \quad \text{Eq. (14)}$$

$$O(Voting\ 1) = O([s \times w] + c) \quad \text{Eq. (15)}$$

$$O(Distributed\ approach\ using\ voting\ 1) = O([m])) + O([s \times w] + c) = O(s \times w) \quad \text{Eq. (16)}$$

Where, m is depth of the decision tree, n is number of nodes in the network, s is the number of sensors, w is the number of assigned weights, c is number of classes.

D. Time Complexity of the Proposed Approach Using the Majority Voting 2

The time complexity of the proposed approach using the majority voting 2 is similar to when majority voting 1 is used with a minor change in the voting part (because it should find a weight corresponding to the currently detected event). The time complexity is therefore calculated by Eq. 17.

$$O(Distributed\ approach\ using\ voting\ 2) = O(m + O[Voting\ 2]) = Max[\,(m), Voting\ 2] \quad \text{Eq. (17)}$$

$$O(Voting\ 2) = O(\,Assiging\ weights + Max\ finding) \quad \text{Eq. (18)}$$

$$O(Voting\ 2) = O([s \times w \times c]) \quad \text{Eq. (19)}$$

$$O(Distributed\ approach\ using\ voting\ 2) = O([m])) + O([s \times w \times c]) = O([s \times w \times c])) \quad \text{Eq. (20)}$$

Where, m is depth of the decision tree, n is number of nodes in the network, s is the number of sensors, w is the number of assigned weights, c is number of classes.

E. Time complexity of the proposed approach using the Majority Voting 3

The time complexity of the distributed approach using the majority voting 3 is similar to when majority voting 2 is used with a minor change because of consolidating similar outputs which are produced by those sensor nodes having the same sensor types. The time complexity is therefore calculated by Eq. 21.

$$O(Distributed\ approach\ using\ voting\ 3) = O([m] + O[Voting\ Technique\ 2]) = Max[(m), Voting\ Technique\ 2]$$ Eq. (21)

$$O(Voting\ 3) = O(Consolidation + Assiging\ weights + Max\ finding)$$ Eq. (22)

$$O(Voting\ 3) = O(s^2 + [s \times w \times c])$$ Eq. (23)

$$O(Distributed\ approach\ using\ 3) = O(m) + O(s^2 + [s \times w \times c])) = O([s \times w \times c])$$ Eq. (24)

Where, m is depth of the decision tree, n is number of nodes in the network, s is the number of sensors, w is the number of assigned weights, c is number of classes.

F. Time Complexity Comparison

Table 4 shows a comparison between time complexities of our different approaches. As it can be seen, our approach using majority voting 1 and reputation-based voting are lighter than the two techniques. The reason that makes majority voting 2 and 3 computationally more intensive is because of assigning more than one weight to each sensor node. This imposed some more comparison and makes the event detection more complex.

Table 4: Time Complexity Comparisons

Approach	Time Complexity
The distributed approach using majority voting 1	$O(s \times w)$
The distributed approach using majority voting 2	$O(s \times w \times c)$
The distributed approach using majority voting 3	$O(s \times w \times c)$
The distributed approach using reputation technique	$O(n^2)$
Where, m is depth of the decision tree, n is the number of sensor nodes in the network, s is the number of sensors, w is the number of assigned weights, c is number of classes.	

VII. CONCLUSIONS AND DISCUSSION OF THE RESULTS

For fast and accurate detection of disastrous events using WSNs, in this paper we propose a distributed event detection technique. Our proposed approach is based on detecting events using decision tree classifiers running on individual sensor nodes and applying a voting to reach a consensus among detections made by various sensor nodes. The motivation behind choosing decision trees is their simplicity and explicit form of expression as if-then-else rules that fulfill the requirements posed by resource limitations of WSNs.

Our experimental results on residential fire datasets show that our approach not only achieves a high detection rate but also has a low computational overhead and time complexity.

ACKNOWLEDGMENT

This work is supported by the EU Seventh Framework Programme and the SENSEI project.

REFERENCES

[1] C. T. Vu, R. A. Beyah, and Y. Li, "Composite Event Detection in Wireless Sensor Networks," in *Proc. of the IEEE International Performance, Computing, and Communications Conference* 2007.

[2] S. Li, Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei1, "Event Detection Services Using Data Service Middleware in Distributed Sensor Networks," *Telecommunication Systems,* vol. 26, pp. 351-368, 2004.

[3] M. L. Segal, F. P. Antonio, S. Elam, J. Erlenbach, and K. R. de Paolo, "Method and apparatus for automatic event detection in a wireless communication system," U. Patent, Ed. USA, 2000.

[4] C. T. Vu, R. A. Beyah, and Y. Li, "Composite Event Detection in Wireless Sensor Networks," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, 2007.

[5] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing,* vol. 10, pp. 18-25, 2006.

[6] M. Bahrepour, N. Meratnia, and P. J. M. Havinga, "Automatic Fire Detection: A Survey from Wireless Sensor Network Perspective," 2008.

[7] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification, and tracking of targets," *Signal Processing Magazine, IEEE,* vol. 19, pp. 17-29, 2002.

[8] W. Xue, Q. Luo, L. Chen, and Y. Liu, "Contour map matching for event detection in sensor networks," in *International Conference on Management of Data*, Chicago, IL, USA 2006.

[9] M. Bahrepour, N. Meratnia, and P. J. M. Havinga, "Use of AI Techniques for Residential Fire Detection in Wireless Sensor Networks," in *AIAI 2009* Greece, 2009.

[10] M. Bahrepour, Y. Zhang, N. Meratnia, and P. J. M. Havinga, "Use of Event Detection Approaches for Outlier Detection in Wireless Sensor Networks," in *ISSNIP 2009* Melbourne, Australia, 2009.

[11] G. Jin and S. Nittel, "NED: An Efficient Noise-Tolerant Event and Event Boundary Detection Algorithm in Wireless Sensor Networks," in *7th International Conference on Mobile Data Management*, 2006.

[12] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers,* vol. 53, pp. 241-250, 2004.

[13] X. Luo, M. Dong, and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *IEEE Transactions on Computers,* vol. 55, pp. 58-70, 2006.

[14] F. Martincic and L. Schwiebert, "Distributed event detection in sensor networks," in *Proceedings of the International Conference on Systems and Networks Communication*, 2006, pp. 43-48.

[15] M. Bahrepour, N. Meratnia, and P. J. M. Havinga, "Fast and Accurate Residential Fire Detection Using Wireless Sensor Networks," *Environmental Engineering and Management,* vol. 9, pp. 215-221, 2010.

[16] M. Marin-Perianu and P. J. M. Havinga, " D-FLER - A Distributed Fuzzy Logic Engine for Rule-Based Wireless Sensor Networks," Springer Verlag, Germany, 2007, pp. 86-101.

[17] A. Khelil, F. K. Shaikh, B. Ayari, and N. Suri, "MWM: A Map-based World Model for Wireless Sensor Networks," in *Autonomics* Turin, Italy 2008.

[18] M. Bahrepour, N. Meratnia, and P. J. M. Havinga, "Sensor Fusion-based Event Detection in Wireless Sensor Networks," in *SensorFusion'09*, IEEE, Ed. Toronto, Canada, 2009.

[19] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 2 edition*: Pearson Education 2003.

[20] Wikipedia, "Decision tree," http://en.wikipedia.org/wiki/Decision_tree.