Use of wireless sensor networks for distributed event detection in disaster management applications

Majid Bahrepour*, Nirvana Meratnia, Mannes Poel, Zahra Taghikhaki and Paul J.M. Havinga

University of Twente,
7500 AE, Enschede, The Netherlands
E-mail: m.bahrepour@utwente.nl
E-mail: n.meratnia@utwente.nl
E-mail: m.poel@utwente.nl
E-mail: z.taghikhaki@utwente.nl
E-mail: p.j.m.havinga@utwente.nl
*Corresponding author

Abstract: Recently, wireless sensor networks (WSNs) have become mature enough to go beyond being simple fine-grained continuous monitoring platforms and have become one of the enabling technologies for early-warning disaster 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; WSNs; situated computing.

Reference to this paper should be made as follows: Bahrepour, M., Meratnia, N., Poel, M., Taghikhaki, Z. and Havinga, P.J.M. (xxxx) 'Use of wireless sensor networks for distributed event detection in disaster management applications', *Int. J. Space-Based and Situated Computing*, Vol. X, No. Y, pp.000–000.

Biographical notes: Majid Bahrepour is a PhD candidate at the Pervasive Group, University of Twente. His current research is focused on data mining, pattern recognition and information discovery in wireless sensor networks. In this research, he is translating various existing machine learning techniques for resource constrained sensor nodes.

Nirvana Meratnia is an Assistant Professor in the Pervasive System Group at the University of Twente. Her research interests are in the area of distributed data management in wireless sensor networks, smart and collaborative objects, ambient intelligence and context-aware applications.

Mannes Poel is an Assistant Professor in the Human Media Interaction Group, University of Twente in The Netherlands. His main research involves applied machine learning for vision-based detection and interpretation of human behaviour and the analysis and classification of EEG-based brain signals.

Zahra Taghikhaki is currently a PhD student at the Pervasive System Research Group of the University of Twente. She received her MSc degree from Iran University of Science and Technology with emphasis on Wireless Sensor Networks. Her research is currently focused on in-network data processing, collaborative event detection and situation awareness in WSN.

Paul J.M. Havinga is a Full Professor and the Chair of the Pervasive Systems Research Group at the Computer Science Department at the University of Twente in the Netherlands. He received his PhD at the University of Twente on the thesis entitled 'Mobile multimedia systems' in 2000, and was awarded with the 'DOW Dissertation Energy Award' for this work. He has a broad background in various aspects of communication systems: on wireless communication, on chip-area network architectures for handheld devices, on ATM network switching, mobile multimedia systems, QoS over wireless networks, reconfigurable computing, and on interconnection architectures for multiprocessor systems.

This paper is a revised and expanded version of a paper entitled 'Distributed event detection in wireless sensor networks for disaster management' presented at INCoS 2010, Thessaloniki, Greece, 24–26 November 2010.

1 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 perspective, 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). WSNs 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 behaviours 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 (Vu et al., 2007).

Resource constraints of the wireless sensor nodes, dynamicity of the deployment area (Li et al., 2004), 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 (Bahrepour et al., 2008).

In this paper, we propose a light-weight and accurate event detection approach for in-network decentralised event detection. The proposed approach uses the decision trees as classifier for the purpose of 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 fulfils the WSNs requirements. The performance of the proposed approach is gauged in terms of detection accuracy and time complexity. We also analyse how various internal parameters of the classification technique influence event detection performance.

2 Related work

Classical research on event detection can be classified into research on

1 pattern matching for known events

2 pattern recognition for unknown events.

Supervised learning techniques are often used for finding events that are similar to predefined event signatures, while unsupervised learning techniques are used for finding hidden patterns.

Events have different meaning in different applications and research domains. Event detection is the process of identifying those sensor readings that do not conform to normal behaviour and model of data and indicate occurrence of an event of interest.

A simple approach to detect events is based on defining thresholds values, with which sensor readings are compared. Threshold-based event detection techniques have been proposed in Segal et al. (2000), Vu et al. (2007) and Werner-Allen et al. (2006). In case of having more than one feature, Vu et al. (2007) propose evaluating different sensor values separately by considering them as an 'atom' or distinct value using a threshold-based classification. For example, if fire event is detected using smoke and temperature sensors, an alarm will be generated when temperature exceeds 30°C and smoke exceeds 100 mg/L. Lim et al. (2007) introduce a generic disaster recovery system using threshold-based event detection technique that may be applicable for various disaster management application.

Defining proper threshold values requires having a good knowledge about the event. Due to the fact that thresholds are defined beforehand and are fixed, threshold-based classification techniques suffer from lack of flexibility and adaptability. To avoid fixed and assumption-based threshold values, Liang and Wang (2005) propose an automatically-selected threshold value approach, in which the threshold value is dynamically calculated using a sliding window technique.

To improve event detection performance in terms of detection accuracy, the new trend is to use pattern recognition techniques. Pattern recognition techniques can be centralised performed in the base station (Li et al., 2002; Xue et al., 2006), local performed in the network (Bahrepour et al., 2009a) or distributed (Bahrepour et al.,

2009b; Jin and Nittel, 2006; Krishnamachari and Iyengar, 2004; Luo et al., 2006; Martincic and Schwiebert, 2006).

Map-based (Khelil et al., 2008), probabilistic-based (Yin et al., 2009), K-nearest neighbourhood-based (K-NN), maximum likelihood-based, and support vector machines-based (SVM) (Li et al., 2002) event detection techniques have been proposed for centralised event detection.

Other techniques based on naïve Bayes (Bahrepour et al., 2009a), feed forward neural networks (Bahrepour et al., 2009a), and SVM (Bahrepour et al., 2010a) have been proposed to detect an event locally on individual sensor nodes. To detect events distributedly in the network techniques based on distributed fuzzy engine (Marin-Perianu and Havinga, 2007), map-based pattern matching (Khelil et al., 2008), feed forward neural networks, naïve Bayes classifiers (Bahrepour et al., 2009c), distributed Bayesian algorithms (Krishnamachari and Iyengar, 2004), voting graph neuron algorithm (Baqer and Khan, 2008), and decision trees (Bahrepour et al., 2010b) have been proposed.

This paper proposes a distributed event detection approach in which decision trees detect events locally on the sensor nodes and a reputation-based voter fuses the local decisions.

3 Decision-tree-based event detection

To fulfil 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 complicated approaches, we will show that simplicity of the decision tress is what WSNs exactly need and our approach can fulfil 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 (Russell and Norvig, 2003; Wikipedia). The inputs to the tree might be either continuous or discrete but the outputs (the decisions) are 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. A typical graphical representation of a decision tree is shown in Figure 1. In a tree, the decision making process starts from the root of the decision tree and propagates down to the leaves.

The main concept of our technique along-side its process and communication model is presented in Figure 2. All sensor nodes run the same decision tree, which is used to detect events locally on the node. Then, the detected events in terms of yes/no or event class type are sent to the voter. The voter applies a reputation-based voting technique to reach a consensus between different decisions made by the nodes.

Figure 1 A typical graphical representation of a decision tree

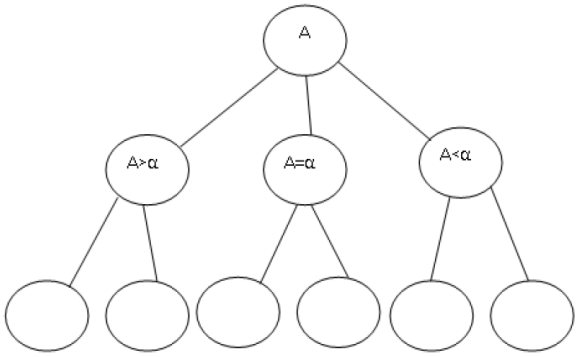
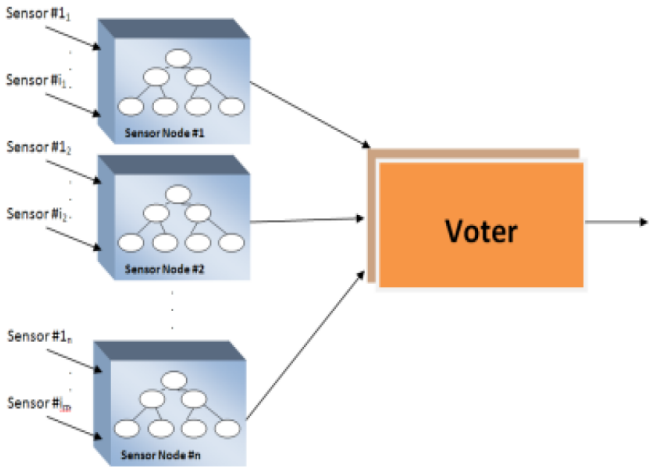


Figure 2 Block diagram of the proposed approach (see online version for colours)



Since sensor nodes send their detected event as a singleton (yes/no or class type) to the voter, the communication between nodes is unidirectional. These two factors make the communication overhead of our proposed approach not high. The rate at which sensor nodes send their data to the voter is variable and is usually based on estimated rate of event occurrences (which is also low).

To show superiority of our reputation-based voting, we additionally investigate three other voting mechanisms based on the classical majority voting.

3.1 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 approach, 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 as a singleton, called detection value (DV), to all other nodes in its neighbourhood. The DVs received from the neighbours will be stored in a table called neighbours detection value table (NDVT). In the next step, each sensor node should judge about its neighbouring sensor nodes by considering itself as the reference. The judgment is accomplished by comparing the difference between detected value of sensor node itself and detected value of the other sensor nodes. If the difference is less than a threshold value θ (representing values that belong to a single class and 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 ($V^{new} = V^{old} - 1$) because its detected value is not in the same class as 'judging' node. 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.

3.1.1 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 (R_i) is obtained based on average value of V^i (positive or negative votes which were given to node *i* by the other sensor nodes) for each sensor node. Then, the average local reputation is multiplied by the weight of sensor nodes calculated using equation (1) to assign global reputation values. The event with the highest reputation weight (W) is the result of the voting procedure. Equation (1) shows how the weights are calculated.

$$W_i = R_i \times Acc_i \tag{1}$$

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 weight of sensor node *i* for event type_q which is calculated based on equation (2).

$$\begin{aligned} Acc_i &= \text{Weight of sensor node}_\text{i} \text{ for event type}_\text{q} \\ &= \sum_{K=1}^{m} \text{Weight of sensor type}_\text{K} \text{ for event type}_\text{q} \end{aligned} \tag{2}$$

3.1.2 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 the sensor node makes 'perfect' decisions, if ($\theta_1 \geq R_i \geq \theta_2$) then the sensor node makes 'fairly good' decisions, and if ($\theta_2 \geq R_i$) then the sensor node makes 'poor' decisions. Then we assign a discrete value for different performances. To do so, 0.5 indicates poor performance, 1 indicates fairly good performance, and 2 indicates perfect performance. Reputation technique 2 is performed using equation (3) based on the performance of each sensor node.

$$W_i = S_i \times Acc_i \tag{3}$$

where W_i is the reputation value corresponding to sensor node *i*, S_i is obtained from equation (4), and Acc_i is the same output of equation (2) (weight of $\text{sensor node}_\text{i}$ for event type_q).

$$S_i = \begin{cases} 2 & \text{if } (R_i \geq \theta_1) \\ 1 & \text{if } (\theta_1 > R_i \geq \theta_2) \\ 0.5 & \text{if } (\theta_2 \geq R_i) \end{cases} \tag{4}$$

θ_1 and θ_2 show how well reputation is (e.g., perfect, fairly good, poor), and is chosen based on application context.

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 5, a number of experiments are conducted and the results are compared.

3.2 Majority voting 1: sensor type-based weighting

In this majority voting technique, we use contribution of each sensor to the event detection process (presented in Table 1) as weight. To do this, we run the same classifier as the event detection classifier with only one sensor type to calculate the contribution of that specific sensor type to the whole event detection process. Then each sensor node receives a weight based on number and types of sensors it has. The weights are calculated using equation (5).

$$\begin{aligned} &\text{Weight of sensor node}_\text{i} \\ &= \sum_{K=1}^{m} \text{Weight of sensor type}_\text{K} \end{aligned} \tag{5}$$

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 equation (6).

$$\begin{aligned} &\text{Weight of sensor type}_\text{k} \\ &= \frac{\text{Detection accuracy using only sensor type}_\text{k}}{\sum_{p=1}^{m} \text{Detection accuracy using only sensor type}_\text{p}} \end{aligned} \tag{6}$$

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

	Fire event detection (in general)	*Flaming fire detection*	*Smouldering fire detection*	*Nuisance detection*
Temperature	19%	26%	20%	13%
ION	16%	2%	20%	20%
Photo	23%	2%	27%	23%
CO	42%	70%	33%	44%

3.3 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. Equation (7) calculates necessary weights for each sensor node.

$$\begin{aligned}&\text{Weight of sensor node}_i \text{ for event type}_q \\ &= \sum_{K=1}^{m} \text{Weight of sensor type}_K \text{ for event type}_q\end{aligned} \tag{7}$$

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 equation (8).

$$\begin{aligned}&\text{Weight of sensor type}_k \text{ for event} \\ &= \text{Event detection for event type}_q \\ &= \frac{\text{Detection accuracy of sensor type}_k \text{ for event}}{\sum_{p=1}^{m} \text{Detection accuracy of sensor type}_p \text{ for event}}\end{aligned} \tag{8}$$

3.4 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 cope with this problem, majority voting 3 gives all sensor nodes having the same sensor types and producing similar event type only one weight. By doing so, we remove redundant weights of sensor nodes having the same sensors and detecting 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.

4 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. The data analysis and simulation of the proposed approach were conducted in MATLAB. In the following subsections the data and experiment methods are described.

4.1 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 smouldering 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, 1,400 data instances were prepared in such a way that 933 instances (2/3) were used for training and 467 instances (1/3) for testing. The dataset contains four sensory data (features) that are temperature, ionisation, photoelectric, and CO. We also perform a calibration procedure to make all the data in the same units.

4.2 Experimental 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 the decision tree is done by using 2/3 of the dataset and testing 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 their 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 the 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.

Various experiments considering different number of sensor nodes and sensor types are conducted. This method of experiment shows robustness of algorithm in case of sensor or node failure, as well as in case of imperfect or lost communication between sensor, which leads to data loss.

The testing phase is conducted by feeding the same instance of data to all sensor nodes then reaching a consensus between results of event detection using one of the voting techniques. The necessary weights for voting part are obtained from contribution of each sensor to the fire detection and are presented in Table 1. It can be seen that CO is contributing the most to the fire detection process. Sensor nodes having more sensor types or the most contributing sensors receive more weight and attention in this study.

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

5 Experimental results

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

Table 2 Network schemas

#	*Availability of sensors*					*Number of sensor nodes*	*Feature*
	Node	*TMP*	*ION*	*Photo*	*CO*		
1	1	✓	✓	×	×	7	Having at least one and at most two sensor types on each sensor node.
	2	×	✓	✓	×		
	3	✓	×	✓	✓		
	4	✓	×	×	✓		
	5	✓	×	×	×		
	6	×	✓	×	×		
	7	×	×	×	✓		
2	1	✓	✓	×	×	3	Having redundant sensor types on each sensor node.
	2	×	✓	✓	×		
	3	✓	×	✓	✓		
3	1	✓	×	×	×	7	Unavailability of CO sensor (as CO is the strongest sensor for fire detection).
	2	×	✓	×	×		
	3	×	×	✓	×		
	4	✓	✓	×	×		
	5	✓	×	✓	×		
	6	×	✓	✓	×		
	7	✓	✓	✓	×		
4	1	✓	×	×	×	8	Having only one CO sensor (as CO is the strongest sensor for fire detection).
	2	×	✓	×	×		
	3	×	×	✓	×		
	4	✓	✓	×	×		
	5	✓	×	✓	×		
	6	×	✓	✓	×		
	7	✓	✓	✓	×		
	8	×	×	×	✓		
5	1–7	✓	×	×	×	8	Having redundant sensor types on each sensor node.
	8	×	×	×	✓		
6	1–5	✓	✓	✓	✓	20	Scaling network to 20 nodes. 25% of the network consists of strong sensor nodes (having all sensor types). And the rest is not very strong (they have only one non-CO sensor type).
	6–10	✓	×	×	×		
	11–15	×	✓	×	×		
	16–20	×	×	✓	×		
7	1–10	×	×	×	✓	100	Scaling network to 100 nodes, having redundant sensor nodes. CO presence in 10% of whole network population.
	11–100	✓	✓	✓	×		
8	1–100	✓	✓	✓	×	100	Scaling network to 100 nodes. All nodes have similar sensor types.
9	1–2	✓	✓	✓	✓	30	Scaling network to 30, having all possible combination of sensor types.
	3–4	×	✓	✓	✓		
	5–6	✓	×	✓	✓		
	7–8	✓	✓	×	✓		
	9–10	✓	✓	✓	×		
	11–12	×	×	✓	✓		

Table 2 Network schemas (continued)

#	*Availability of sensors*					*Number of sensor nodes*	*Feature*
	Node	*TMP*	*ION*	*Photo*	*CO*		
9	13–14	✓	×	×	✓	30	Scaling network to 30, having all possible combination of sensor types.
	15–16	✓	✓	×	×		
	17–18	×	✓	✓	×		
	19–20	✓	×	✓	×		
	21–22	×	✓	×	✓		
	23–24	×	×	×	✓		
	25–26	✓	×	×	×		
	27–28	×	✓	×	×		
	29–30	×	×	✓	×		
10	1–50	✓	✓	✓	✓	50	Scaling network to 50 by having the strongest possible sensor nodes.

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

Table 3 Results of the distributed approach

Net. architecture #	*Technique*	*Event detection accuracy*	*Standard deviation*
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	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
6	Rep. technique I	99.57%	0.27
	Rep. technique II	99.57%	0.27
	V. technique #1	99.14%	0.63
	V. technique #2	99.25%	0.38
	V. technique #3	98.95%	0.52

Table 3 Results of the distributed approach (continued)

Net. architecture #	*Technique*	*Event detection accuracy*	*Standard deviation*
7	Rep. technique I	92.38%	0.8
	Rep. technique II	92.38%	0.8
	V. technique #1	85.53%	9.71
	V. technique #2	87.77%	5.04
	V. technique #3	87.1%	13.54
8	Rep. technique I	91.61%	1.08
	Rep. technique II	91.61%	1.08
	V. technique #1	88.57%	6.23
	V. technique #2	90.57%	4.5
	V. technique #3	90.63%	5.16
9	Rep. technique I	98.72%	0.30
	Rep. technique II	98.5%	0.53
	V. technique #1	98.9%	0.49
	V. technique #2	98.93%	0.3
	V. technique #3	98.85%	0.59
10	Rep. technique I	99.64%	0.2
	Rep. technique II	99.64%	0.2
	V. technique #1	99.12%	0.45
	V. technique #2	99.07%	0.31
	V. technique #3	99.05%	0.39

Based on Table 3, we can generally conclude that reputation-based voting techniques work better than majority voting techniques. However, in fifth experiment, reputation-based voting is not working well because there is only one sensor node having the most contributing sensor (CO), and there are seven sensor nodes having only temperature (not very contributing to fire detection according to Table 1). Then, the seven sensor nodes because of their quantity receive higher weights in reputation-based voting and perform event detection less accurate. One can also conclude that the reputation-based voting performs best when redundant nodes in the network are not many.

Another conclusion to make is related to the number of sensor nodes in the network and its effect on detection accuracy. Comparing the first experiment with seventh and eighth, it can be seen that increase of number of nodes does not necessarily improve the detection accuracy. However, presence of the most contributing sensors has the strongest effect on detection accuracy.

We additionally compare the proposed technique with another distributed event detection approach presented in Bahrepour et al. (2009c), in which neural networks are used as both the local event detector and voter. We use network scheme number 10 and reputation-based voting as one of the best achieved simulation results. Then the same dataset is given to the distributed neural network (Bahrepour et al., 2009c) which has the same sensor nodes and sensor types as network scheme number 10. Table 4 shows the result of this comparison.

Table 4 Comparing the proposed technique with distributed neural network approach

Approach	*Event detection accuracy*
Rep. technique I and II	99.64%
Distributed neural network (Bahrepour et al., 2009c)	96%

Source: Bahrepour et al. (2009c)

As it can be seen in Table 4, the reputation-based technique outperforms the distributed neural network approach in terms of event detection accuracy.

6 Time complexity analysis

Our aim is to investigate applicability of computationally intensive ML techniques for resource-limited WSNs. 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)

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.

6.1 Time complexity of the decision tree

The order of the decision tree appraisal is a function of the depth of decision tree and equation (9) presents the time complexity:

$$O(\text{Local approach}) = O(\text{Decision tree appraisal}) \tag{9}$$

$$O(\text{Local approach}) = O(m) \tag{10}$$

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 decreases the classification accuracy in most of circumstances, as well.

6.1.1 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. The time complexity is calculated by equation (11).

$$\begin{aligned} &O(\text{Distributed reputation}) \\ &= \max[O(\text{Distributed reputation}) \\ &\quad + O(\text{process on the node}) + O(\text{reputation voting})] \end{aligned} \tag{11}$$

$$\begin{aligned} &O(\text{Distributed reputation}) \\ &= Max[O(m) - O(n(n-1)) \\ &\quad + O((n(n-1)) + n + c)] \end{aligned} \tag{12}$$

$$O(\text{Distributed reputation}) = O\left(n^2\right) \tag{13}$$

where m is depth of the decision tree, n is the number of sensor nodes in the network and c is the number of classes.

6.2 Time complexity of the proposed approach using the majority voting 1

In the distributed approach, sensor nodes detect events in parallel using decision trees. Therefore, the order of whole classification part is $O(m_1 + m_2 + \ldots + m_n) = O(m)$; 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 equation (14).

$$\begin{aligned} &O(\text{Distributed approach using voting 1}) \\ &= O([m]) + O([\text{Voting 1}]) \\ &= Max[(m),\ \text{Voting 1}] \end{aligned} \tag{14}$$

$$\begin{aligned} &O(\text{Voting 1}) \\ &= O(\text{Assigning weights + Max finding}) \end{aligned} \tag{15}$$

$$O(\text{Voting 1}) = O([s \times w] + c) \tag{16}$$

$$\begin{aligned} &O(\text{Distributed approach using voting 1}) \\ &= O([m]) + O([s \times w] + c) \\ &= O(s \times w) \end{aligned} \tag{17}$$

where m is the 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 and c is the number of classes.

6.3 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 equation (18).

$$\begin{aligned} &O(\text{Distributed approach using voting 2}) \\ &= O(m + O[\text{Voting 2}]) \\ &= Max[(m),\ \text{Voting 2}] \end{aligned} \tag{18}$$

$$\begin{aligned} &O(\text{Voting 2}) \\ &= O(\text{Assigning weights + Max finding}) \end{aligned} \tag{19}$$

$$O(\text{Voting 2}) = O([s \times w \times c]) \tag{20}$$

$$\begin{aligned} &O(\text{Distributed approach using voting 2}) \\ &= O([m]) + O([s \times w \times c]) \\ &= O([s \times w \times c]) \end{aligned} \tag{21}$$

where m is the 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 and c is number of classes.

6.4 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 equation (22).

$$\begin{aligned} &O(\text{Distributed approach using voting 3}) \\ &= O([m] + O([\text{Voting technique 2}]) \\ &= Max[(m),\ \text{Voting technique 2}] \end{aligned} \tag{22}$$

$$\begin{aligned} &O(\text{Voting 3}) \\ &= O(\text{Consolidation + Assigning weights} \\ &\quad + \text{Max finding}) \end{aligned} \tag{23}$$

$$O(\text{Voting 3}) = O\left(s^2 + [s \times w \times c]\right) \tag{24}$$

$$\begin{aligned} &O(\text{Distributed approach using voting 3}) \\ &= O(m) + O\left(s^2 + [s \times w \times c]\right) \\ &= O([s \times w \times c]) \end{aligned} \tag{25}$$

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.

6.5 Time complexity comparison

Table 5 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 has lower complexity than the other 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 requires more comparisons and makes the event detection more complex.

Table 5 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	$O(n^2)$

Notes: n is the number of sensor nodes in the network, s is the number of sensors, w is the number of assigned weights and c is the number of classes.

7 Communicational overhead

As presented in Figure 2 the communication model of our technique is unidirectional, in which sensor nodes send their decision about occurrence of events to the voter (data fuser). The frequency of sending data to the voter is application dependent and is based on estimated frequency of occurrence of the event (which is low). As each sensor node sends only a singleton (showing decision about occurrence of events) and not its entire sensor data to the voter, and sending data is unidirectional towards the voter, the communication overhead is minimal.

8 Parameter study

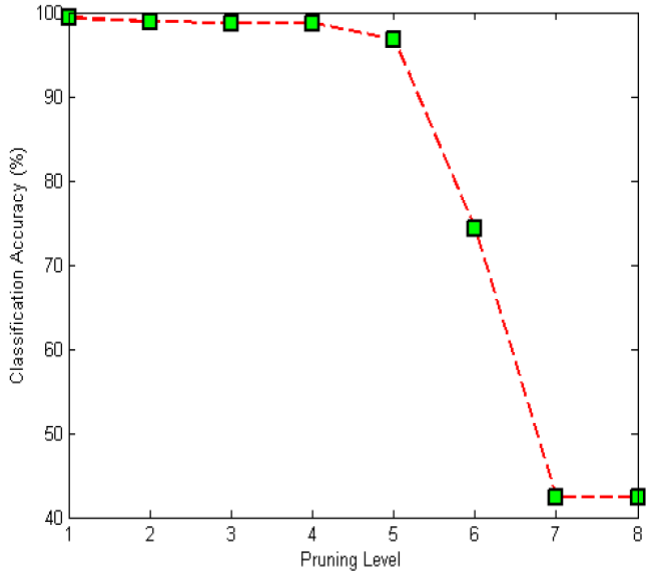
8.1 Pruning

Decision trees have no parameter because the whole tree is made during the training phase and there is no need to choose parameters. However, after the training phase, the tree can be pruned and some less-contributing branches can be removed. Figure 3 shows how pruning a tree can affect accuracy of a decision tree using our fire dataset.

According to equation (8) and (9) the depth of decision tree has direct effect on time complexity of the approach. Therefore, pruning can be considered as a technique to reduce time complexity. However, by pruning a decision tree, accuracy rate is also decreased.

As a conclusion, decision tree can be pruned to make a faster yet less-accurate classifier.

Figure 3 Pruning and its effect on final detection accuracy for fire dataset (see online version for colours)



8.2 *Parameter study for reputation-based voting technique*

There are some parameters involved in reputation-based voting technique that affect event detection accuracy. In this section we investigate three major parameters of the reputation-based voting approach. These parameters are

1 reputation update time intervals

2 number of neighbours

3 presence of sensor types.

Moreover, the way reputation is calculated, whether it is done centrally or in a distributed manner affects the time complexity of the proposed approach.

8.2.1 *Distributed vs. centralised reputation calculation*

Finding the trustworthiness or reputation degree of the sensor nodes can be accomplished either in a distributed way (by engaging neighbours of the 'being judged node') or in a centralised manner (in a base station). In some circumstances the base stations have not enough power to obtain reputation degree of each sensor node, therefore, a distributed fashion can come handy, which also provides load balancing for reputation values calculation between nodes. However, if base stations have enough power in terms of computing power, memory capacity, battery and radio bandwidth, the reputation calculation process becomes computationally lighter.

As a result of this change, the $n(n-1)+n$ term, which belongs to the local communication among the neighbouring nodes in order to find out the reputation value, is omitted from equation (12). Every sensor node only needs to report its detected values to the base station. Therefore, the number of required communications of each sensor node is diminished to 1 from $(n-1)+1$ and as a result of this change the energy dissipation of each node extremely decreases.

8.3 *Reputation update time interval*

In our proposed method, each sensor node sends every detected value to all its neighbours and receives their corresponding values in each time period. Although, it results in more reliability but it causes quick energy drain of the sensor nodes. In contrast, each sensor node can store its detected values in a table which should be sent to other neighbours after a certain time period. It also needs then to receive the neighbours' tables after a specific time period. As time interval becomes longer, both communication overhead and reliability are reduced.

Figure 4 shows the performance of the proposed method for ten different runs in terms of detection accuracy while varying the time interval for updating reputation values. As it is expected, the detection accuracy increases by decreasing the time interval at the expense of consuming more energy.

Figure 4 Effect of reputation update time interval on detection accuracy (see online version for colours)

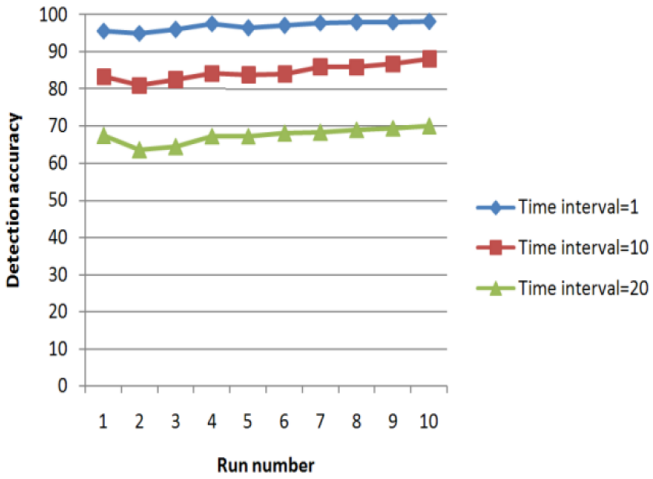
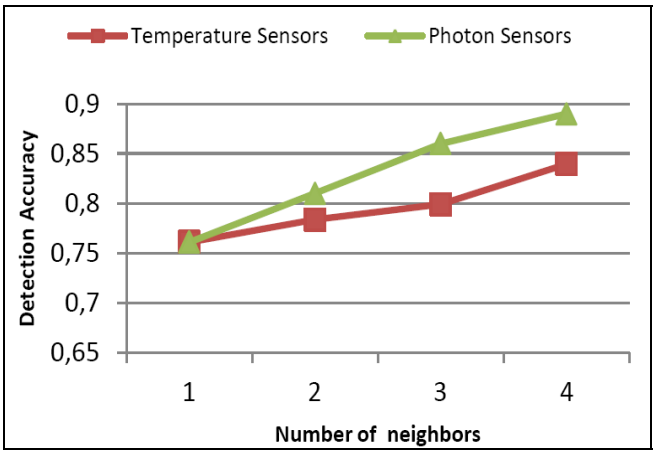


Figure 5 Detection accuracy for different number of the neighbours (see online version for colours)



8.3.1 Number of neighbours

Number of neighbours around a sensor node has direct impacts on the detection accuracy as it affects the reputation and trustworthiness of a given node. Figure 5 presents the impact of increasing number of neighbours for two different sensor types (temperature sensor, photon sensor). It can be seen that detection accuracy is improved by increasing the number of neighbours for a given sensor type.

8.3.2 Presence of sensors

In addition to number of nodes, presence of the most contributing sensors affects detection accuracy. As Table 3 shows, when the most contributing sensor (CO, in fire detection scenario) is not available, the detection accuracy drops off. This is due the fact that, having many sensor nodes missing important sensor(s) results in a situation in which sensor nodes cannot correctly detect an event. As it can be seen from Figure 5, increasing number of neighbours of a node with a contributing sensor (photon sensor in this case) leads to better detection accuracy compared to increasing number of neighbours of node with less contributed sensor (temperature sensor for example).

9 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 technique to reach a consensus among detections made by various sensor nodes. We proposed three majority-based voting techniques and two reputation-based voting techniques. The experimental results show that reputation-based voting approaches perform well in absence of high degree of redundant nodes in the network.

The motivation behind choosing decision trees and aforementioned voting techniques is their simplicity, low computational costs and high accuracy which fulfil 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.

The future works includes implementation of the proposed approach on Crossbow Telos sensor nodes.

References

Bahrepour, M., Meratnia, N. and Havinga, P.J.M. (2008) 'Automatic fire detection: a survey from wireless sensor network perspective', in Technical Report TR-CTIT-08-73, Centre for Telematics and Information Technology, University of Twente, Enschede, ISSN 1381-3625.

Bahrepour, M., Meratnia, N. and Havinga, P.J.M. (2009a) 'Use of AI techniques for residential fire detection in wireless sensor networks', in *AIAI 2009*, Greece.

Bahrepour, M. et al. (2009b) 'Use of event detection approaches for outlier detection in wireless sensor networks', in *ISSNIP 2009*, Melbourne, Australia.

Bahrepour, M., Meratnia, N. and Havinga, P.J.M. (2009c) 'Sensor fusion-based event detection in wireless sensor networks', in *SensorFusion '09*, IEEE, Editor, Toronto, Canada.

Bahrepour, M., Meratnia, N. and Havinga, P.J.M. (2010a) 'Fast and accurate residential fire detection using wireless sensor networks', *Environmental Engineering and Management*, Vol. 9, No. 2, pp.215–221.

Bahrepour, M. et al. (2010b) 'Distributed event detection in wireless sensor networks for disaster management', in *International Conference on Intelligent Networking and Collaborative Systems, INCoS 2010*, IEEE Computer Society, Thessaloniki, Greece.

Baqer, M. and Khan, A.I. (2008) 'Event detection in wireless sensor networks using a decentralised pattern matching algorithm', White Paper.

Jin, G. and Nittel, S. (2006) 'NED: an efficient noise-tolerant event and event boundary detection algorithm in wireless sensor networks', in *7th International Conference on Mobile Data Management*, IEEE Computer Society.

Khelil, A. et al. (2008) 'MWM: a map-based world model for wireless sensor networks', in *Autonomics '08*, Turin, Italy.

Krishnamachari, B. and Iyengar, S. (2004) 'Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks', *IEEE Transactions on Computers*, Vol. 53, No. 3, pp.241–250.

Krishnamachari, B. and Iyengar, S. (2004) 'Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks', *IEEE Transactions on Computers*, Vol. 53, No. 3, pp.241–250.

Li, D. et al. (2002) 'Detection, classification, and tracking of targets', *Signal Processing Magazine*, IEEE, Vol. 19, No. 2, pp.17–29.

Li, S. et al. (2004) 'Event detection services using data service middleware in distributed sensor networks', *Telecommunication Systems*, Vol. 26, No. 2, pp.351–368.

Liang, Q. and Wang, L. (2005) 'Event detection in sensor networks using fuzzy logic system', in *EEE Intl. Conference on Computational Intelligence for Homeland Security and Personal Safety*, Orlando, FL.

Lim, Y-s. et al. (2007) 'A fire detection and rescue support framework with wireless sensor networks', in *International Conference on Convergence Information Technology*, IEEE Computer Society.

Luo, X., Dong, M. and Huang, Y. (2006) 'On distributed fault-tolerant detection in wireless sensor networks', *IEEE Transactions on Computers*, Vol. 55, No. 1, pp.58–70.

Marin-Perianu, M. and Havinga, P.J.M. (Eds.) (2007) *D-FLER – A Distributed Fuzzy Logic Engine for Rule-Based Wireless Sensor Networks*, Springer Verlag, Germany, pp.86–101.

Martincic, F. and Schwiebert, L. (2006) 'Distributed event detection in sensor networks', in *Proceedings of the International Conference on Systems and Networks Communication*.

Russell, S.J. and Norvig, P. (2003) *Artificial Intelligence: A Modern Approach*, 2nd ed., Pearson Education.

Segal, M.L. et al. (2000) 'Method and apparatus for automatic event detection in a wireless communication system', U. Patent, Editor, USA.

Vu, C.T., Beyah, R.A. and Li, Y. (2007) 'Composite event detection in wireless sensor networks', in *Proc. of the IEEE International Performance, Computing, and Communications Conference*.

Werner-Allen, G. et al. (2006) 'Deploying a wireless sensor network on an active volcano', *IEEE Internet Computing*, Vol. 10, No. 2, pp.18–25.

Wikipedia, Decision Tree, available at http://en.wikipedia.org/wiki/Decision_tree.

Xue, W. et al. (2006) 'Contour map matching for event detection in sensor networks', in *International Conference on Management of Data*, Chicago, IL, USA ACM New York, NY, USA.

Yin, J., Hu, D.H. and Yang, Q. (2009) 'Spatio-temporal event detection using dynamic conditional random fields', in *International Joint Conference on Artificial Intelligence*, Pasadena, California, USA.