Composite Event Detection in Wireless Sensor Networks

Chinh T. Vu, Raheem A. Beyah and Yingshu Li
Department of Computer Science, Georgia State University
Atlanta, Georgia 30303
{chinhvtr, rbeyah, yli}@cs.gsu.edu

Abstract

*Sensor networks can be used for event alarming applications. To date, in most of the proposed schemes, the raw or aggregated sensed data is periodically sent to a data consuming center. However, with this scheme, the occurrence of an emergency event such as a fire is hardly reported in a timely manner which is a strict requirement for event alarming applications. In sensor networks, it is also highly desired to conserve energy so that the network lifetime can be maximized. Furthermore, to ensure the quality of surveillance, some applications require that if an event occurs, it needs to be detected by at least k sensors where k is a user-defined parameter. In this work, we examine the Timely Energy-efficient k-Watching Event Detection problem (*TEKWED*). A topology-and-routing-supported algorithm is proposed which constructs a set of detection sets that satisfy the short notification time, energy conservation, and tunable quality of surveillance requirements for event alarming applications. Simulation results are shown to validate the proposed algorithm.*

1 Introduction

Event alarming is an effective function of Wireless Sensor Networks (WSNs) which is employed in many applications such as meteorological hazard detection, earthquake-tsunami alerting, and enemy detection in battle fields. In the literature, the most popular scenario is as following: all the sensors periodically send their data to an information-processing center, *e.g.*, a base station (BS), a conclusion is then made at the BS to decide whether a pre-defined event has happened based on the reported data. We name this scenario as "data collection". Another more efficient scenario is "data aggregation", where data is processed at some nodes in-network before being forwarded. However, these methods are not suitable for some realtime applications, especially for emergency alarming applications, where the alarm is stringently required to be announced in a timely manner. The drawbacks of most of the existing methods are that the realtime requirement is not taken into account, and the amount of the exchanged data may be huge which causes large energy consumption. In addition, at a BS, the received data need to be further analyzed to obtain a conclusion which delays the alarm to be timely announced.

Because of its "emergency" characteristic, an emergency alarming application intrinsically differs from data collection/aggregation applications. First, the interested information is an answer to a question which can be derived by a set of predicates, not from the raw data in the form of numerical values. For example, for fire alarming applications, the users are not interested in the exact temperature or the smoke density of the monitored area. Instead, the users expect the quick answer to the concise question "is there any fire in the monitored area?". Second, to guarantee accuracy and reliability for emergency alarming applications, a conclusion indicating the happening of an event should not be decided only based on one property of the event. For example, the event *fire* is a fusion of multiple sensed values of multiple different attributes, *i.e.*, the occurrence of fire should satisfy some conditions such as $temperature > 100^oC \ AND \ smoke > 100mg/L$, rather than a simple condition $temperature > 100^oC$ or $smoke > 100mg/L$ alone. Any change in either $temperature$ or $smoke\ density$ that makes $temperature > 100^oC$ or $smoke > 100mg/L$ true is an *atomic event*. The event that is a combination of several atomic events is a *composite event*, *e.g.*, the composite event *fire* is represented as $temperature > 300^oC \ AND \ smoke > 100mg/L$. The formal definition of an event is given in Section 3.1. In this paper, we investigate how to efficiently detect events for emergency alarming applications using WSNs. To detect a composite event, a number of sensing devices with different sensing components need to be involved and local operations and collaborations are conducted to give out the answer. To conserve energy, only the answer is sent to the BS.

The basic idea of our proposed scheme is illustrated in Fig.1, where an area is monitored by a WSN. There is a

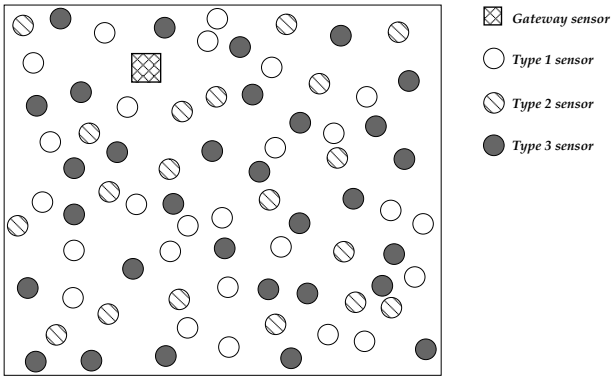

Figure 1. Event query in a WSN

gateway node (or several gateway nodes) which is responsible for making a conclusion and reporting it to the users if an event happens. This gateway node is properly selected and every sensor in the network has a chance to serve as a gateway node in order to balance the energy consumptions. To detect events based on different properties, multiple types of sensors are deployed. For example, in Fig.1, *type 1, type 2, and type 3* sensors are used for temperature, smoke density, and light monitoring, respectively. An event E is defined with a compound propositional function as following:

$$E = \mathcal{F}(P_1(x), ..., P_n(x)) \quad (1)$$

where $P_1(x)$ through $P_n(x)$ are predicates, $\mathcal{F}$ is a function of Boolean algebra operators such as '$\wedge$', '$\vee$' or '$\neg$'. For example, an event $fire$ can be defined as $Fire = P_1(x) \wedge P_2(x) \wedge P_3(x)$, where $P_1(x)$ denotes the predicate $temperature > 300^oC$, $P_2(x)$ denotes the predicate $smoke > 100mg/L$, and $P_3(x)$ denotes the predicate $light > 500cd$.

The event E and threshold values can be disseminated to gateway and non-gateway nodes by the BS at the initial phase or pre-installed in each sensor. Only the gateway nodes have the information about an event E. Each non-gateway node only knows the threshold values of its monitored properties. During the network operation time, once a sensor detects that the current sensed value is over the threshold of its monitored property, it sends one bit '1' instead of the sensed value to a gateway node. If a gateway node receives a '1', it checks if the compound propositional function which defines an event E derives a $TRUE$ value. If so, it immediately sends a warning to the BS. It is important to emphasize that a sensor need not periodically report its rawly sensed data. Instead, it only reports the predicate '1', that is, it notifies the gateway node only when its sensed value reaches the threshold and refrains from sending data for all other cases.

For WSNs, energy conservation is always a primary objective. Additionally, to ensure that events can be properly reported by inclined-to-failure WSNs, the proposed scheme is also required to provide a certain level of fault-tolerance. Different from many traditional solutions for the coverage problem where connectivity is assumed and is ignored, in event alarming applications the connectivity is extremely important since an event needs to be alerted on time. To meet these requirements, our proposed scheme divides the sensors into non-disjoint subsets and each subset can conduct the event alarming process mentioned above with a user defined fault-tolerance level. Instead of requiring all the sensors to be active all the time, only one subset is responsible for the event alarming task at any time. In this way, energy can be conserved and the network lifetime can be extended. Our novel scheme has the following contributions and characteristics:

1. No significant amount of data is sent to the BS, thus each sensor can naturally conserve more energy to extend network lifetime. Further, reducing network traffic has a good effect on lowering radio interference. Also, the energy consumption among sensors is well balanced.
2. Decisions are locally made at special sensors, namely gateway nodes, then only particular conclusions, *e.g.*, the ones that specify the occurrence of an interested event, are reported to the BS, so that the users can obtain valuable information in a timely manner.
3. The pattern definition of an event may consist of multiple properties instead of a single one and can be defined by users.
4. By sending a warning from a gateway node to the BS, the BS can know where the event happens (with the assumption that the BS knows all the nodes' positions).
5. Even if any limited number of sensors concurrently fail, the BS can still be properly warned in a timely manner if any interested event happens.
6. The connectivity is guaranteed among sensors in the current detection set (defined in Section 3.1) by using a topology and routing control scheme.

The remainder of this paper is organized as follows. Section 2 presents some related work on the event detection problem. In Section 3, the problem statement and some related definitions are provided. The algorithm is described in detail in Section 4. The simulation results are shown in Section 5. Finally, Section 6 gives the conclusion.

2 Related Work

The problems of event detection and notification are investigated in the distributed system research fields. In [3],

a framework for both simple event and composite event detection is developed using distributed collaboration of sensors which may have different sensing capabilities. An application subscribes an interested event (it may be atomic or composite) with a corresponding location, and the proposed protocol builds an *event-based tree* (EBT) and the data in the form of predicates will then be collected along this tree. Both atomic and composite event detection protocols are proposed in this work. The protocol for the former is relatively straightforward. For a composite event, the protocol maintains a counter for each atomic event to count the number of the nodes which can sense it and the node is added to the cluster until the counter for each atomic event is greater than a predefined threshold. The sensing predicate is sent through the reverse path to the interested application which subscribes the event and will decide if the event happened. In [4], the issue of how to make the decision from predicates is addressed. Some existing event notification systems for sensor networks and their disadvantages are mentioned in the paper. Since an atomic event frequently changes states, an automaton can be used to combine the state changes of atomic events to form the final states and those final states help decide if the desired event occurs. The drawback of an automaton is its complexity, *i.e.*, the number of its states may be exponential of the number of the atomic events and there are probably infinite ways to reach a state. Moreover, the time overlap of state changes is hard to handle by an automaton. The issue of disseminating events to the users (or sensor nodes) also attracts much attention from researchers. In [1], the hop-tree-based routing scheme is introduced with the objective of finding a short path between the event source and the query nodes. First, each sensor is marked with a hop-level number specifying the length in hops from it to a randomly-chosen root node. The hop-levels of a node's neighbors are used to specify direction (up in the tree or down in the tree) to route the events and queries. In [2], directed diffusion - a data-centric paradigm - is developed. First, the BS broadcasts the interest under the form of attribute-value pairs to all the sensors. Each sensor stores the *interest, timestamp, gradient* and some other information in its cache. If the sensor's sensing data matches the interest, the sensor uses the gradient field to send the data back to the BS. Neither [1] or [2] consider the energy balancing issue.

The above works do not consider some of the special characteristics of WSNs. We should be aware that WSNs have limited communication ability, data-centric feature, limited power, limited computation ability, large number of nodes, huge deployment area and infinite sensing data streams. To the best of our knowledge, no research work in WSNs has been conducted for event detection and alarming considering the stringent requirement of delivering a timely warning. Different from all the above works, we partition the set of sensors into a number of non-disjoint subsets, each of which is referred as a "*detection set*", such that each subset can solely detect atomic events for all the predicates of a composite event. Atomic events are reported to a gateway node, which can easily contact with the BS. The gateway node then decides if a composite event happens and notifies the BS. The most relevant work to ours is [3]. Nonetheless, the work in [3] concentrates on system issues rather than network issues. It does not consider the energy consumption and it discovers only one detection set for each sub-region. Besides, the proposed protocol in [3] requires the sensors' locations. Oppositely, energy is one of our foremost concerns since it is a key issue in sensor networks. Also, our algorithm does not require the sensors' position information. We further consider other strict requirements in sensor networks such as fault-tolerance, connectivity, topology and routing control for the resulting detection sets.

3 The TEKWED Problem

In this section, we formally define the TEKWED problem and the following are some preliminary definitions and notations.

3.1 Preliminaries

Definition 1 *(*Detection set*) A subset of sensors which jointly accomplish the event detection and alarming task.*

Definition 2 *(*Notification time*) Notification time is the summation of the time for all the members within a detection set to report the atomic events to the gateway, the time for the gateway to make a decision, and the time for a gateway to notify the BS that an event happens.*

From [4], we adopt the following definition for an *event*:

Definition 3 *(*Event*) An event is a change of a real-world state.*

Then, a k-watched atomic and a k-watched composite event are defined as follows:

Definition 4 *(*k-watched atomic event*) An atomic event is said to be k-watched by a set of sensors D if at any time this event occurs at any point within the interested area, at least k sensors in the set D can detect this occurrence.*

Definition 5 *(*k-watched composite event*) A composite event is said to be k-watched by a set of sensors D if every atomic event forming that composite event is k-watched by set D.*

The following are some notations that we use in our algorithm.

- N: The number of sensors.
- S: The set of sensors. *i.e.*, $S = \bigcup_{i=1..N} s_i$.
- k: The user-defined fault-tolerance level.
- r: The number of atomic events whose combination forms the composite event of interest. r depends on the pattern definition of the composite event.
- m: The number of detection sets.
- $D_j(j = 1..m)$: The j^{th} detection set.
- $t_j(j = 1..m)$: The assigned active time of set D_j.
- $\sigma_l(l = 1..r)$: The l^{th} atomic event.
- Σ: The composite event. *i.e.*, $\Sigma = \mathcal{F}(\sigma_1, .., \sigma_r)$.
- $\chi_i(i = 1..N)$: The sensor s_i's current contribution.

3.2 Problem Definition

Generally, in k-watching event detection, at least k sensors know the occurrence of an event. So the composite event is ensured to be detected even when any $k-1$ sensors concurrently fail. Also, k-watching helps the BS to be more quickly notified of events than the case where some atomic events are watched by only one sensor. In a primitive and standard form, the k-watching problem can be defined as follows:

Definition 6 (k-watching problem) *Given a set of sensors S, a monitored area A, and a composite event Σ which is a combination of r atomic events $\sigma_l,\ l = 1..r$, find a subset D of S such that every σ_l is k-watched by the set D.*

However, practical applications always require several other network constraints. In event detection and alarming applications, the most important issue is to timely report the event to the event consumer, *e.g.*, the BS. Obviously, to ensure the messages are properly routed, a path to a gateway node must be well maintained, *i.e.*, the connectivity for the detection set must be provided. Additionally, conserving energy while accomplishing tasks is important. Thus, minimizing the notification delay and energy consumption are our primary concerns in this work. Since atomic event detection and alarming is a special case of composite event detection and alarming, in this work we only consider the latter. Formally, our concentration can be summarized as following:

Definition 7 (Timely Energy-efficient k-Watching Event Detection - **TEKWED**) *Given a set of sensors S, a monitored area A, and a composite event Σ which is a combination of r atomic events $\sigma_l,\ l = 1..r$, find a set of non-disjoint connected subsets (detection sets) $D_j,\ j = 1..m$ of S, and decide their corresponding active duration and subset masters (gateway nodes) such that:*

1. *The composite event Σ is k-watched by $D_j,\ j = 1..m$ at any time.*
2. *The network lifetime is maximized.*
3. *For each detection set, the notification time is minimized.*

4 Construction of Detection Sets

In this section, we describe our proposed algorithm for constructing detection sets in a WSN in detail.

4.1 Assumptions

- Each node has different sensing abilities. That is, each sensor may be equipped with more than one sensing components and both the numbers and types of those sensing components may be different among sensors. For example, a sensor can sense light intensity and/or smoke density, while its neighbor can sense temperature and/or pressure.
- For each type of sensing component, a sensor is equipped with no more than one sensing component. For example, a sensor has only one temperature sensing component and/or one pressure sensing component.
- All of a sensor's sensing components turn on or off simultaneously.
- Compound function $\mathcal{F}$ and predicates are diffused to all the sensors in the network deployment phase or a mechanism as discussed in [2] is used to accomplish this task.
- Sensors may have different communication ranges and different initial battery supplies.

4.2 Algorithm description

The BS is responsible for constructing a set of detection sets by executing the algorithm shown in Algorithm 1. During the active time of a detection set, the sensors in this detection set route their messages to the gateway node based on the topology and routing information provided by the BS. Basically, the messages are routed to the gateway node along the constructed BFS tree for the current detection set. In this paper, we interchangeably use the term *sensor* and *node*. The $Construct - Leaves$ function at line 9 is given in Algorithm 2.

The detection sets construction algorithm starts by choosing a node to be the gateway - a special node in charge of making the decision about the occurrence of the composite event and notifying the BS if it happens. The gateway can simply be any node with enough residual energy. The next step is to construct a set of connected BFS-like trees rooted at the gateways, and the nodes in each of these trees form a detection set. For each atomic event σ_l, we maintain a counter c_l recording the number of the currently needed sensors who can detect σ_l for the current detection set to provide k-watching for σ_l. The initial value of c_l is k for each event σ_l. A sensor may be equipped with several sensing components that can monitor different σ_l. For a sensor s_i, a sensing component $component_{i,l}$ is called a *helpful*

Algorithm 1 : Detection-Sets-Construction(S, k)

1: m=0.
2: **while** $S \neq \phi$ **do**
3: $T = \phi$. Set all the counters c_l to k.
4: Color all the nodes WHITE.
5: Choose a node gw close to the event consumer and have more energy as a gateway; gw.Color = BLACK.
6: $T = \{gw\}$.
7: */* Discover detection set */*
8: **while** *at least one counter* > 0 **do**
9: $\mathcal{L}$ = Construct-Leaves(S, T)
10: (Re)Calculate the contribution of each node in $\mathcal{L}$.
11: Color all the nodes with contribution of 0 RED and remove them out of $\mathcal{L}$.
12: **if** $\mathcal{L} == \phi$ **then break**;
13: Sort $\mathcal{L}$ in descending order of contributions.
14: **while** $\mathcal{L} \neq \phi$ **do**
15: Remove the node ρ from the top of list $\mathcal{L}$.
16: Add node ρ into T.
17: ρ.*Color = (*ρ.*Parent is BLACK)*? BLACK : GREEN
18: Decrease all the ρ's correlated counters by 1.
19: **if** *a counter == 0* **then**
20: *all counters == 0*? **goto** line 25 : **goto** line 10
21: **end if**
22: **end while**
23: **end while**
24: */*Update the subset and the sensors' energy*/*
25: **if** *any counter* > 0 **then**
26: Remove T from S. */* T is isolated */*
27: **else**
28: **if** *there exists a GREEN node* **then**
29: **foreach** *GREEN node* ρ **do**
30: κ = ρ.Parent
31: **while** κ *is RED* **do**
32: κ.Color = BLACK
33: Add κ to T.
34: κ = κ.Parent
35: **end while**
36: **end for**
37: **end if**
38: m=m+1
39: $gw_m = gw$; $D_m = T$ */* A new detection set */*
40: Assign t_m the smallest lifetime of a node in D_m.
41: Recalculate residual energy of sensors in D_m.
42: Remove from S the sensors who ran out of energy.
43: **end if**
44: **end while**
45: **return** m, $\{D_j,\ gw_j,\ t_j\}_{j=1..m}$

sensing component for counter c_l if it can monitor σ_l and the current value of c_l is greater than 0, and c_l is called a

Algorithm 2 : Construct-Leaves(S, T)

Input: A set of sensors S and a tree T.
Output: $\mathcal{L}$ - the list of all the children of T's leaves

1: Construct a list $\mathcal{L}$ consisting of all the WHITE neighbors of T's leaves.
2: /* Assign parent for each node in $\mathcal{L}$*/
3: **foreach** *node* ρ *in* $\mathcal{L}$ **do**
4: **if** *any neighbor of* ρ *is BLACK or GREEN* **then**
5: ρ.Parent = the BLACK/GREEN node with the least number of children.
6: **else**
7: ρ.Parent = the RED node with the longest lifetime.
8: **end if**
9: **end for**
10: **return** $\mathcal{L}$.

correlated counter for component $component_{i,l}$. At any point of time, a color in the following list, is used to represent a sensor's state:

- $WHITE$: a sensor has not been considered.
- $BLACK$: a sensor has already been added to a detection set and is connected to the gateway through other nodes in the current detection set.
- RED: a useless sensor (the one whose all correlated counters c_l are 0) has been considered, but has not been added to a detection set.
- $GREEN$: a sensor has been added to a detection set but its parent is not a $BLACK$ node, thus it is not connected to the gateway through other nodes in the current detection set.

At the beginning of a detection set construction, all the nodes are $WHITE$ except the $BLACK$ gateway node gw, thus the tree T contains only gw. Denote $\mathcal{L}$ as a list of all of T's leaves' WHITE neighbors, which intuitively is the set of nodes in the next level in the BFS tree of the current T's leaves' level.

Our algorithm works in a greedy manner on a sensor's attribute named contribution χ to find a sensor that can be added to the tree. For a sensor s_i, its contribution χ_i can be determined depending on some parameters such as its residual energy, the energy needed to transmit a message to its parent node, and the number of its helpful sensing components. χ_i can be formulated as follows:

$$\chi_i = f(e_i, d_i) \times \frac{h_i}{sc_i} \tag{2}$$

where:

- $f(e_i, d_i)$ is a function to calculate s_i's lifetime depending on its current residual energy e_i and its current communication range d_i. This function is further discussed in detail in Section 5.

- h_i is the number of s_i's helpful sensing components.
- sc_i is the number of all the sensing components that s_i is equipped with.

Eq. 2 can easily be extended to take any other parameters into account. Notice that the contribution of a sensor becomes 0 when h_i=0, *i.e.*, when all the correlated counters for all its sensing components are 0, it becomes a useless sensor (for current detection set).

The algorithm tries to construct as many detection sets as possible. At each iteration of our algorithm's main loop, a temporary variable T is used to store the current tree being constructed. When the tree is completely built, it becomes a new detection set. T is gradually constructed by adding the node in $\mathcal{L}$ who has the biggest contribution. For other nodes in $\mathcal{L}$ having contribution of 0, they are colored RED. Each time a node is added to T, it is removed from $\mathcal{L}$ and all of its correlated counters are decreased by 1. Also, the node added to T is colored $BLACK$ if it can connect with the gateway node through other $BLACK$ nodes. If it cannot, *i.e.*, its parent is a RED or $GREEN$ node, it becomes $GREEN$. Each time any counter becomes 0, all the remaining sensors in $\mathcal{L}$ need to recalculate their contribution χ_i, since their h_i values may change. T's construction process finishes when *a)* all the counters reach 0, *i.e.*, sensors in T can now provide k-watching for the composite event or *b)* there exists no T's neighbors. For the later case, remove all the sensors in T from S (line 26, algorithm 1) since T is isolated from the other sensors in S. For the former case, to guarantee connectivity, some RED nodes need to be added to make $GREEN$ nodes connected to the gateway, which is accomplished by the block of code from line 28 to line 37. When T is completely built, it contains only $BLACK$ and $GREEN$ nodes. It is then assigned an active time t which is the smallest lifetime of a sensor in T.

The algorithm keeps constructing more trees using the above process until no more detection sets can be discovered. Finally, the algorithm returns the constructed detection sets with their active durations and the corresponding gateway nodes.

It is worth emphasizing several special points in our heuristic:

- Since the compound proposition is really simple and it is easy to derive the result, any ordinary sensor can make the decision from the reported predicates' values (the '1's sent by the sensors within its detection set) without any difficulty, that means a special type of sensor to work as a gateway node is not necessary.
- While constructing $\mathcal{L}$ (Algorithm 2), we explicitly specify the parent for each node in $\mathcal{L}$. Thus a topology for each detection set is as also established. Furthermore, when a sensor needs to report an atomic event, it only needs to forward that report to its assigned parent. By that mechanism, no addition routing protocol is needed.
- In our algorithm, we trade off between the energy consumption and the notification time while building the tree. By using a BFS tree, the notification time is supposed to be smaller. However, if the DFS is used for the BLACK nodes instead of using RED nodes to connect GREEN nodes to BLACK nodes, the energy consumption should be smaller (since the number of intermediate RED nodes is minimized).

Theorem 1 *The algorithm ensures that a composite event is k-watched by every detection set and all the detection sets are connected sets.*

Due to page limitation, we intentionally omit the proof for this theorem.

4.3 An example

As mentioned, a composite event can be understood as the combination of some atomic events, *e.g.*, high temperature, dazzling light and dense smoke indicate a *fire* event. Fig. 2 illustrates an example of a fire alarming system. Each sensor is equipped with one or several sensing components where *1* is for temperature, *2* is for light and *3* is for smoke sensing component. The bold number above each sensor is the sensor ID and the list beside it is the list of its equipped sensing components. Notice that in Fig. 2b, each node is assigned a unique parent and in Fig. 2c, node 5 is changed to $BLACK$ and is added to the detection set with the purpose of connecting $GREEN$ node 7 and 10 to $BLACK$ node 3. Thus, if node 10 wants to report an event to the gateway node 1, it can send that report along the path: 10-7-5-3-1.

5 Simulation

5.1 Simulation setting

The function to calculate sensor s_i's lifetime mentioned in Eq. 2 can be computed as follows:

$$f(e_i, d_i) = \frac{e_i}{Tx_i + Sx_i} \tag{3}$$

where:

- e_i is the current residual energy (in mJ).
- Sx_i is the energy needed to sense an event. Since we do not consider the coverage problem in this work, for the simulation part, we assume that Sx_i is proportional to the number of the sensing components sc_i equipped on s_i, and each sensing component spends $10(mJ/unit\ of\ time)$ when it is turned on. Thus, $Sx_i = 10 * sc_i\ (mJ/unit\ of\ time)$.

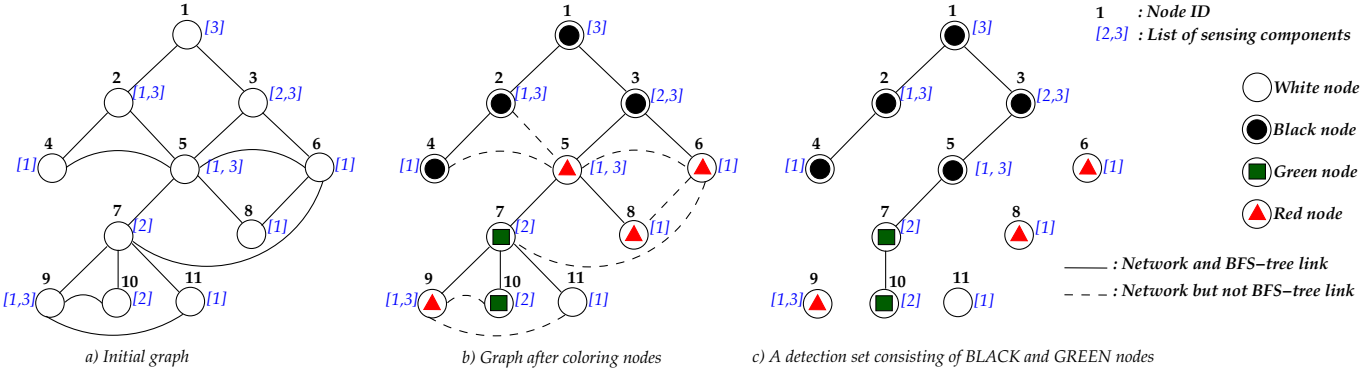

Figure 2. The construction of a detection set with $k = 3$, $r = 3$. Node 1 is the gateway node.

- Tx_i is the energy needed to transmit a message which is a function of the sensor's communication range d_i. The following model is widely adopted in the literature: $Tx_i = a \times d_i^{\alpha} + \beta$ where a, α and β are constants, $2 \leq \alpha \leq 4$ and a is usually set to 1 [5].

Table 1 presents our simulation setting.

Table 1. Simulation Settings

Area size	$100m \times 100m$	r	4
Communication range	$15\ m$	α	2
Initial energy	$20 \rightarrow 30J$	β	0

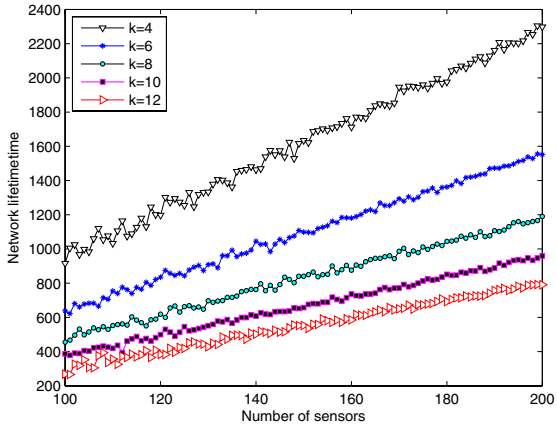

Figure 3. Network lifetime

5.2 Simulation Results

In this section, we evaluate the efficiency of our heuristic through conducting simulations measuring the network lifetime and notification time. We also compare the network lifetimes for different values of k and different composite event pattern definitions. For each measurement, we run the simulations on 50 different completely-randomized networks and report the average results.

Fig. 3 shows the network lifetime with the assumption that a composite event (continuously) happens all the time, so the sensors in each detection set have to keep reporting events all the time, which is the worst case in practice. This assumption makes it easier to illustrate our algorithm's performance. That means the practical network lifetime resulted from our algorithm must be longer than what is shown in Fig. 3. As can be observed, the network lifetime is relatively proportional to the ratio $\frac{N}{r \times k}$. This effect is understandable since the bigger the value of the level of fault-tolerance k or the number of the atomic events r, the more sensors need to be involved in the event detection task, thus the more energy that the network consumes in a unit of time, hence the smaller the network lifetime. On the other hand, the larger the number of the sensors, the bigger the number of the detection sets, hence the longer the network lifetime.

In Fig. 4, we measure the notification time (as being defined in Def. 2). Because we have no specific BS, we omit the time from the gateway node to the BS. The larger the number of sensors, the smaller the height of the tree, thus the notification time is consequently smaller. On the contrary, the higher the level of fault-tolerance, the larger the number of sensors involved in detecting the event, consequently the notification time increases. Notice that Fig. 4 shows the total of the time for all the members of a detection set to route their message to the gateway node by using the routing path created by our algorithm without any aggregation. For each atomic event, the payload of the message sent by each sensor is only a couple of bits - $\lceil log_2 r \rceil$ bits (depending on the composite event pattern definition) for the type of atomic event, and only 1 bit for

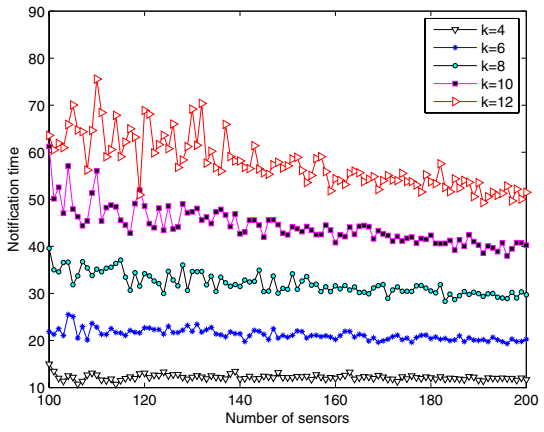

Figure 4. Notification time

the value (which is always 1 showing that the atomic event happens). Thus, it is highly practical to employ a data aggregation mechanism such as [6] in reporting the event to the gateway. The notification time would greatly diminish if such a mechanism is utilized. The notification time is actually the time that the gateway node needs to know for sure that the composite event indeed occurs, *i.e.*, the time for the gateway to receive k reports for each atomic event. However, the gateway can be aware of the occurrence of a composite event if it receives only one report for each atomic event. Meaning that the gateway may be notified of the occurrence of the event in much shorter time. To measure this kind of time, we introduce a new simulation parameter named $average\ notification\ time$ which is $\frac{Notification\ time}{k}$. This value is from 2.5 to 6.5 in our simulations. It is small enough for an event to be warned in a timely manner as required by $TEKWED$.

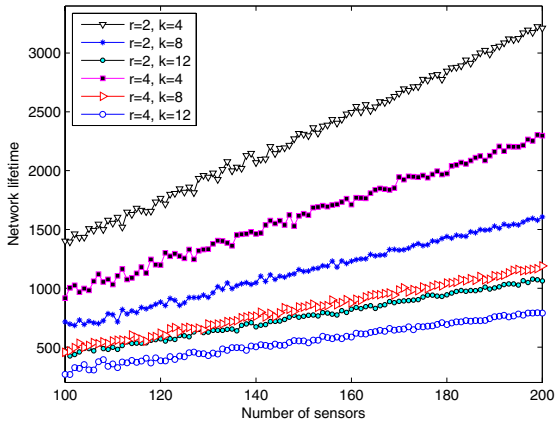

Figure 5. Network lifetime for different r, k

Fig. 5 compares the network lifetimes for several values of k and r. As can be seen, with the same value of k, the network lifetime decreases when the value of r increases. Similarly, with the same value of r, the network lifetime also decreases when the value of k increases. The effect can be reasoned by the same explanation we have made for Fig. 3.

6 Conclusion

In this paper, we examine the TEKWED problem for composite event detection and alarming in WSNs. We introduce a simple scheme for detecting events and timely delivering warnings in WSNs. Based on that scheme, we propose an energy-efficient algorithm that considers topology and routing control of the network and takes into account fault tolerance. Simulation results are presented and analyzed to evaluate our algorithm's efficiency. In this work, only one type of composite event is considered. In the future, we will investigate how to detect different (related) types of composite events and send warnings in a timely manner.

References

[1] T.W. Chim. *Along & Across Algorithm for Routing Events and Queries in Wireless Sensor Networks*. International Symposium on Intelligent Signal Processing and Communication Systems, 2005, pp.725-728.

[2] C. Intanagonwiwat, R. Govindan and D. Estrin. *Directed diffusion: a scalable and robust communication paradigm for sensor networks*. ACM MobiCom, 2000, pp.56-67.

[3] D. Janakiram, A.V.U.P. Kumar and A.M. Reddy V. *Component Oriented Middleware for Distributed Collaboration Event Detection in Wireless Sensor Networks*. MPAC05, November 28 - December 2, 2005.

[4] K. Römer and F. Mattern. *Event-based systems for detecting real-world states with sensor networks: a critical analysis*. DEST Workshop on Signal Processing in Sensor Networks at ISSNIP, December 2004, pp.389-395.

[5] I. Stojmenovic and X. Lin. *Power-Aware Routing in Ad Hoc Wireless Networks*. IEEE transactions on parallel and distributed systems, 2001.

[6] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta. *A Low-Latency and Energy-Efficient Algorithm for Convergecast in Wireless Sensor Networks*. GLOBECOM, 2003, 22(1):3525–3530.