TED: Efficient Type-based Composite Event Detection for Wireless Sensor Network

Steven Lai, Jiannong Cao
Internet and Mobile Computing Lab
The Hong Kong Polytechnic University
Kowloon, Hong Kong
Email: {csylai, csjcao}@comp.polyu.edu.hk

Xiaopeng Fan
Shenzhen Institutes of Advanced Technology
Chinese Academy of Sciences
China
Email: xp.fan@siat.ac.cn

***Abstract*—Although there are several works on providing event-based services in pervasive environment or WSN, most of them have not considered composite event detection in an energy-efficient fashion. Composite events consist of multiple primitive events with temporal and spatial relations and are much more difficult to manage. Because of the resource constraints in WSN, existing event detection algorithms may not be suitable for WSN when energy efficiency is considered. In this paper, we propose TED (Type-based composite Event Detection), a distributed composite event detection algorithm. The essential idea of TED is type-based event fusion, where some sensor nodes are selected as fusion points. Then lower-level events will be fused on these fusion points for detection of higher-level composite events. Each composite event type is assigned to certain fusion point for detection so that the composite events may be detected in-network instead of at the sink. Event fusion with minimum energy cost is an NP-complete problem. We propose a distributed randomized algorithm to solve the problem. We analyze the energy efficiency of TED to show both its effectiveness and efficiency. By carrying out both simulation and real world experiments on TED, we show that TED can reduce the energy cost by 10-20% in event-based WSN applications compared with naïve event detection mechanism where the event relations are not considered.**

I. Introduction

Composite event [1] detection in WSNs is required in many applications such as health care [2], smart building [3] and intelligent transportation system [4]. For instance, in an intelligent transportation system, we may define traffic jam as an event when there are certain number of cars waiting on a road. Such kind of event may come from many sub-events such as the number of the vehicles on a road and the speed of the vehicles. Furthermore, the speed of the vehicles may come from the events of each individual vehicle. All these sub-events must satisfy certain spatial or temporal relations in order to indicate an occurrence of a composite event.

In general, composite event detection in WSN can be considered as a special type of predicate detection and events in WSN applications have two special properties. First, events in many WSN applications have short life spans. For instance, object tracking usually involve events that occur as the object passes the network. After the object has exited the monitoring region, the primitive events related to that object will usually become useless. Second, events in WSN may have strong locality. For example, consider a surveillance application where the events are defined to characterize intrusions into certain region, such events usually happen at the the entrance or the exit of the region.

Moreover, energy is usually considered as a crucial issue for WSN. It is more preferable to detect composite events in-network instead of gathering all the primitive events at sink and performing centralized event detection. As a result, event detection in WSN quite different from event detection in active databases [5] when issues such as network dynamics and resource constraints are considered.

As an important paradigm for distributed communication, pub/sub is especially suitable for event-based applications due to its decoupling [6]. There are mainly three kinds subscriptions used in pub/sub system: topic-based subscription, content-based subscription and type-based subscription. In particular, composite events are usually defined in type-based pub/sub system by specifying event types and their relations. Certain existing works [7] have addressed the issue of defining and subscribing such kind of composite events for WSN. However, how to perform energy efficient composite event detection has not been fully addressed in these works.

In this paper, we formally address the problem of composite event detection for WSN. We propose TED (Type-based Event Detection), a distributed algorithm for efficient composite event detection. The main idea of TED is to detect composite events based on their types. Then assign each composite event type to a certain node called event fusion point so that the composite events can be efficiently detected. The contribution of this paper can be summarized as follows:

1) We formulate the problem of type-based composite event detection.
2) We propose TED, a distributed type-based composite event detection algorithm.
3) We analyzed the performance of TED and conducted both simulations and experiments to show the performance of TED in event-based WSN applications.

The rest of the paper is organized as follows: Section II presents the background for our composite event detection approach. Section III reviews related works. Section IV presents our system model and problem formulation. Section V describes TED in details. Section VI shows the performance of

TED through analysis, simulation and experiments. Section VII concludes the paper.

II. Background

A. Event Types and Subscriptions

Each event contains a list of attributes which are the actual data obtained from the sensors. Users can subscribe events with subscriptions which specify event types and event filters. Different event types can have relations among each other. The relations may also be expressed as special type of filters that operate on multiple event types. Listing 1 shows how a composite event type is defined [7]. The event type 'CompTemp' requires two sub-events of type 'AvgTemp'. The two sub-events must satisfy certain temporal constraints in order to indicate the occurrence of the composite event.

Listing 1: Example of a composite event

```
1  Event CompTemp {
2  } on {
3      AvgTemp e1 and
4      AvgTemp e2
5  } where {
6      e2.time-e1.time=600
7  }
```

Once the subscriptions are defined, it will be disseminated into the network so that sensor nodes start to detect the corresponding events. In a pub/sub system, if an event is detected to match the subscribed event type, it will be delivered to the sink to notify the subscriber.

B. Type-based Event Detection

In our approach to event detection, we make use of some special nodes called event fusion points. These nodes take the responsibility of composite event detection. The event fusion points may have some extra capabilities in terms of processing power or storage though it's not compulsory. For each event fusion point, it has an event detection framework as shown in Figure 1. In such a framework, events are stored in a buffer and associated with types. Each event type has a corresponding filter so that the fusion points can decide if the event occurs or not. The event filters consist of some predicates on the event attributes. Event detection based on our model involves three steps:

1) Detect an event either locally or receive an event from other nodes. Then store it in the buffer.
2) Look up for the corresponding filter for that event.
3) Evaluate the event against the filter.
4) If the event is evaluated to have happened, make a decision on where to send the event.

The event detection framework in Figure 1 is essentially type-based event detection because the predicates are grouped according to the event types. In addition, compared with topic or content-based approach, type-based event detection is more suitable for WSN-based applications. Because of the resource constraints in WSNs, it is usually not efficient to send all the primitive events to the sink node for centralized event detection. Instead, events should be detected in-network if possible. Composite events in WSNs may have multiple levels and higher level events cannot be detected until its sub-events have been detected. Therefore, in-network event detection must take the event types into consideration in order to detect the events.

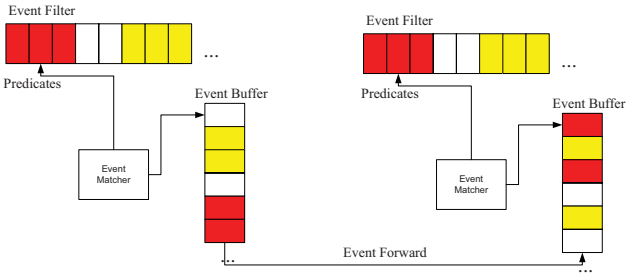


Fig. 1: Type-based event detection framework

III. Related Works

Event detection is one of the basic issues considered in many WSN-based applications and systems. Earlier works include query-based event definition and detection [8]. However, queries may not be sufficient to describe events with complicated spatial and temporal relations [9]. In addition to event queries, the pub/sub paradigm has also been used. Existing pub/sub based works for WSNs, however, primarily consider the detection of primitive events where each event is usually treated separately and events don't have relations between each other. [10] is a pub/sub system built on top of directed diffusion [11]. The sink node will first broadcast interest and the source nodes will set up the gradients. When the events are detected, they will be delivered through reinforced paths. Mires [12] is a pub/sub middleware for WSN. It makes use of the message-oriented communication paradigm provided by TinyOS. First, nodes will advertise their available topics using a multi-hop routing protocol. Then, the sink will broadcast the subscription and finally, nodes will be able to publish the events to the sink.

More recently, certain works such as [7], [13] have been proposed for composite event definition and detection. The primary focus of [7] is on the system design so that different composite event detection algorithms can be easily integrated into the system. In addition to an event language, [13] also discusses how to reliably detect composite events in a pervasive environment. However, energy-efficient composite event detection problem has not been well addressed in the context of WSN.

In this paper, we study the composite event detection problem by considering certain characteristics of the events in WSN such as spatial or temporal relations [14], relatively short event lifetime and data redundancy [15]. We believe composite event detection is important for many WSN applications because many of these applications are event-based in nature.

More work needs to be done in order to efficiently detect composite events.

IV. The Composite Event Detection Problem

A. System Model

We consider the network as a graph $G = (N, A)$ where each node represents a sensor node and each edge represents a communication link. For each $a_n \in A$, it has a weight W_n associated with it.

The subscriber provides a finite set of event types $E = \{e_1, e_2, \cdots\}$. For each $e_n \in E$, the subscriber defines a set of attributes $e_n \rightarrow attr_n$ which reflect certain real world phenomenon.

The subscriber also provides a finite set of event relations $R = \{r_1, r_2, \cdots\}$ where each $r_n \in R$ represents the mapping of one or more sub-event types $e_1, e_2 \cdots \in E$ to a composite event type $e_3 \in E$, denoted as $r_n(e_1, e_2, \cdots) = e_3$. One of the event type $e_s \in E$ is subscribed by the subscriber.

We have a set of primitive event types $E_{primitive} \subseteq E$ such that $r_n(e_1, e_2, \cdots) = e_n$ where $e_n \in E_{primitive}$ and $e_1, e_2, \cdots \in E$. For each primitive event of type $e'_n \in E_{primitive}$, it will be detected by a node $n_i \in N$. We use the message cost as the event detection cost for each event type $e_n \in E$, denoted as $cost(e_n)$.

B. The General Problem Formulation

Given:

- A network $G = (N, A)$
- A set of event types E with relation R
- A cost function $cost(e_n)$ for $e_n \in E$

Find:

- For each event type $e_n \in E$, when an event of this type is happens, find a subset of nodes $V_n^r \subseteq V$ which are involved in detecting the event.

Objective:

- Minimize the total energy consumption:

$$\sum_{i=i}^{n} cost(e_i)$$

Theorem 1. *The composite event detection problem is NP-complete.*

We may reduce this general composite event detection problem to Steiner tree problem since given a set of events, the minimum energy cost can be obtained if we connect these events through the Steiner nodes in the network. The detailed proof is omitted for the sake of brevity.

V. TED in Action

In this section, we propose TED, a distributed type-based composite event detection algorithm for WSN. The essential idea of TED is that after each sub-event is detected, the nodes will at first forward the detected events randomly to some nearby fusion points in the hope that at least some of them will be able to detect events at lower cost. When the composite events are detected, the fusion points will first check to see if the source nodes have already selected any fusion point. If not, it will flood some feedback in the network so that the source node will get it and other nodes can also use such feedbacks as 'hints' when they need to forward the events. By collecting different feedbacks from different fusion points, the sensor nodes will choose the best one according to the cost. If the sub-events occur again, the nodes will be able to forward the detected events based on the feedback so that the cost could likely be reduced.

A. Algorithm Input

In TED, the set of event fusion points $N_f \subseteq N$ are preselected. We will discuss how to select the fusion points in an optimal way in the latter part of the section. Therefore, each node will play two possible roles: normal node or event fusion point. Normal nodes will need the following data structure for the algorithm:

- Event filter table $table_f$: this table stores the filters for each event type. $table_f \rightarrow filter_n$ denotes the filter for event type e_n.
- Fusion point routing table ($table_r$): this table defines the routing to each fusion point $n_i \in N_f$. $table_r \rightarrow n_i \rightarrow parent$ denotes the parent node to reach fusion point n_i.
- Event forwarding table ($table_e$): this table defines for each event type $e_n \in E$, the corresponding fusion point for the it. $table_e \rightarrow e_n \rightarrow fp$ denotes the fusion point for event type e_n.

The fusion points will also have the same data structure of the normal nodes for the algorithm. In addition, they will have an additional table $table_m$. This table temporarily stores the events collected from other nodes. For each of the entries it has the following contents:

- e_n^i: the i^{th} event of type e_n
- $cost$: the detection cost for the event e_n^i
- $flag$: the flag (to be described in the algorithms) for the event e_n^i

B. TED for Normal Nodes

Since the event detection starts from primitive events, the normal nodes will run Algorithm 1 after detecting a primitive event e_n^i of type e_n. For each event type, it has three possible states: fpUnknown, fpIndicated and fpSelected. Initially, all the event types are fpUnknown because the sensor node does not know which fusion point is the best to forward the event. The flag will be updated upon the reception of feedbacks from the fusion points. More specifically, if the event is detected at the fusion point n_i, the fusion point will flood the feedback with cost and event source included so that the nodes can update their corresponding flags. The update is based on the detection cost.

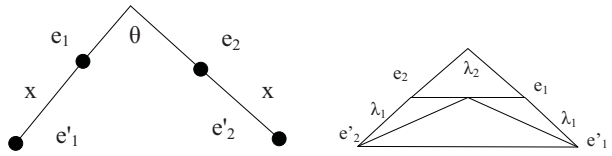
Upon the detection of event e_n^i, the node will first check if there is already a fusion point assigned to it. If so, the event will simply be forwarded to that fusion point. Otherwise, the node will choose k closest fusion points randomly and then forward the events to them.

Algorithm 1 TED for normal nodes

Input: evaluate(e_n^i, $table_f \rightarrow filter_n$)==True
  if $table_e \rightarrow e_n \rightarrow flag \neq fpUnknown$ **then**
    $toForward = table_e \rightarrow e_n \rightarrow fp$
    Set $e_n^i \rightarrow flag = table_e \rightarrow e_n \rightarrow flag$
    Forward e_n^i to $table_r \rightarrow fp \rightarrow parent$
  else
    if $table_e \rightarrow e_n \rightarrow flag \neq fpUnknown$ **then**
      Select $k-1$ nearest fusion points $N_k \in N_f$
      $N_k = N_k \bigcup \{table_e \rightarrow e_n \rightarrow fp\}$
    else
      Select k nearest fusion points $N_k \in N_f$
    end if
    for each $n \in N_k$ **do**
      forward e_n^i to $table_r \rightarrow n \rightarrow parent$
    end for
  end if
  if $e_n^i \rightarrow flag = fpSelected$ **and** $e_n^i \rightarrow timeout == True$ **then**
    $table_e \rightarrow e_n \rightarrow flag = fpIndicated$
  end if
Input: feedback of event type e_n from $n_i \in N_f$
  $entry = table_e \rightarrow e_n$
  if $e_n \rightarrow source == self$ **and** ($entry \rightarrow flag \neq fpSelected$ **or** $entry \rightarrow flag == fpSelected$ **and** $entry \rightarrow cost < e_n \rightarrow cost$) **then**
    $entry \rightarrow flag = fpSelected$
    $entry \rightarrow fp = n_i$
  else if $entry \rightarrow cost < e_n \rightarrow cost$ **then**
    $table_e \rightarrow e_n \rightarrow flag = fpIndicated$
  end if

Algorithm 2 TED for fusion points

Input: e_n^i from node $n_i \in N$
  for all $e_j \in E$ **do**
    if e_n is a subevent of e_j **then**
      result = evaluate e_j with e_n^i
      if result==True **then**
        detected (e_j)
        $e_j \rightarrow cost = e_j \rightarrow cost + e_n^i \rightarrow cost$
        $e_j \rightarrow source = e_j \rightarrow source \bigcup e_n^i \rightarrow source$
        feedback (e_j)
      end if
    end if
  end for
Input: expiry time of e_n^i
  detected (e_j)



(a) Event distribution (b) Fusion point distance

Fig. 2: Fusion points selection

C. TED for Event Fusion Points

When the fusion point receives e_n^i from a node, it will first wait a period of time until the expiry time of the event to check for other events for possible matches. If no match is found during this period, the fusion point will still use Algorithm 1 to further forward the events to other fusion points. The pseudo code is shown in Algorithm 2.

The function 'detected' is the place where Algorithm 1 is invoked. Upon the detection of any composite event, the fusion point will also send the feedbacks to the network.

D. Determine the Re-selection Probability

While the previous sections outline the algorithm for TED, we still need to decide how often the nodes should switch to another fusion point in order to cope with the event dynamics and network topological changes. We use exponential distribution as the event probability distribution because of its memoryless property. More specifically, for each composite event, the distance between each of its sub-events to any point in the network follows an exponential distribution as follows:

$$f(x) = \lambda_1 e^{-\lambda_1 x}$$

In addition to the distance between events, the direction of events that happen in different rounds will also affect the selection probability. The angle between any pair of related events also satisfy an exponential distribution as follows:

$$f(\theta) = \lambda_2 e^{-\lambda_2 \theta}$$

Both x and θ are shown in Figure 2a where the events e_1, e_2 are detected before and e_1', e_2' are detected. For simplicity, we use distance to measure the cost for one node to reach another. Let d be the average distance between any point in the deployment region to the closest sensor node. As shown in Figure 2b, for a composite event that has two sub-events, originally the event is fused at n_1. Then, for the next detection of e_1' and e_2', if the fusion point is still the original one, then the cost will be no less than:

$$cost_1 = \sqrt{(\lambda_1)^2 + (\frac{\lambda_1}{2})^2 + (\lambda_1)^2 cos(\frac{\pi + \lambda_2}{2})}$$

On the other hand, if a new fusion point is selected, then the cost will be no more than:

$$\begin{aligned} cost_2 =& 2(\frac{\lambda_1}{2sin\frac{\lambda_2}{2}} + \lambda_1) sin\frac{\lambda_2}{2} + d \\ =& \lambda_1(1 + 2sin\frac{\lambda_2}{2}) + d \end{aligned}$$

The condition to select a new fusion point will be:

$$\begin{aligned} & cost_1 \geq cost_2 \\ & cost_1^2 \geq cost_2^2 \\ & (\lambda_1)^2(\frac{5}{4} + cos(\frac{\pi + \lambda_2}{2}) \geq \\ & \lambda_1^2(1 + 2sin\frac{\lambda_2}{2})^2 + d^2 + 2\lambda_1(1 + 2sin\frac{\lambda_2}{2})d \end{aligned} \quad (1)$$

Here d is decided by the node density and can be estimated once we know the deployment area and the number of nodes in the deployment area. Therefore, given $f(x)$, $f(\theta)$ and d, we can use certain numeric methods such as generalized gradient search to find the values for θ and x such that Equation 1 is satisfied while minimizing the probability for switching:

$$\begin{aligned} P_{switch} &= F(x > x*)F(\theta > \theta*) \\ &= \int_{x*}^{\infty} \lambda_1 e^{-\lambda_1 x} dx \int_{\theta*}^{\infty} \lambda_2 e^{-\lambda_2 \theta} d\theta \\ &= e^{-\lambda_1 x*} e^{-\lambda_2 \theta*} \end{aligned}$$

Once the probability is obtained, after each event detection, there will be a probability of P_{switch} that the event will switch to another fusion point. This is done by making the fusion point broadcast a message in the network so that all nodes can delete the corresponding event type assignment to that fusion point.

E. Fusion Point Deployment

Because our distributed algorithm is based on certain nodes in the network that acts as event fusion points to detect the events, in this section, we discuss how to select such fusion points in order to efficiently detect the events. To give an answer without losing generality, we use the following deployment model:

- The entire network is divided into a set of equally sized regions.
- Within each region, we deploy the same number of event fusion points.

Such deployment model is suitable if the user has no prior knowledge on where the events would happen. After calculating the optimal deployment strategy, the users can make use of it in two ways:

- Even deployment: after the sensor deployment, the users can deploy additional sensor nodes as fusion points in the network.
- Random deployment: before the deployment, the user can calculate how many fusion points are needed in the network and mix them with normal nodes to deploy them randomly.

We will use square for calculating the optimal deployment strategy in this paper. The optimal deployment strategies with regions of other types of shape may be also be obtained in a similar fashion. Suppose we divide the whole region of area A into squares of size $s \times s$. Then on average, each sensor node can find a fusion point at a distance of:

$$\begin{aligned} r &= \int_0^1 D(t)dt \\ &= \int_0^1 \frac{2}{3}\sqrt{c^2t^2 + (b^2 - a^2 - c^2)t + a^2}dt \\ &= \frac{c}{6}[u(1 + v^2) + \frac{1}{2}(1 - u^2)(1 - v^2)ln(\frac{u - 1}{u + 1})] \end{aligned} \quad (2)$$

where,

$$\begin{aligned} c &= \frac{s}{2} \\ u &= \frac{\sqrt{2} + 1}{2}s \\ v &= \frac{\sqrt{2} - 1}{2}s \end{aligned} \quad (3)$$

In order to determine the optimal deployment strategy, we also need to know the event probabilistic distribution. We use the same exponential distribution model as above.

The cost introduced by TED mainly consists of three parts: forwarding cost, feedback cost and detection cost. Initially, upon the detection of primitive events, the nodes will randomly forward the events to k closest fusion points.

$$cost_{forward} = r \times k$$

Here k is determined by the event distribution such that after forwarding different sub-events to the fusion points, there may be some overlapping fusion points for different sub-events. Therefore, k is defined as follows:

$$k = (\frac{\lambda_1}{r} + 1)^2 \quad (4)$$

When the events are detected at the fusion points, feedback will be sent to the event sources so that the sensor nodes can later forward the events to them and the cost will be reduced. For simplicity of analysis, we assume the fusion points will simply flood the feedback in the network. Therefore, the feedback cost is:

$$cost_{feedback} = |N| \times k$$

The detection cost is the message cost for all sub-events to be forwarded to a fusion point so that the composite event may be detected. If we have two events e_1 and e_2, the minimum event detection cost will be detecting the events on the line segment that connects the two events. However, we may not find a fusion point on the line segment, so in order to find a fusion point that can minimize energy cost, we should choose a point that lies on the bisector of the line segment (the detailed proof is omitted for brevity). Similar to Equation 2 and 3, the average detection cost will be:

$$cost_{detect} = 2 \times \int_0^{arctan\frac{2r}{\lambda_1}} \frac{\lambda_1}{2cosx}dx$$

Since each node needs to know how to reach the fusion points when forwarding is needed, there is overhead for maintaining such information. Similar to many existing routing

λ_1	Expected location between the events
λ_2	Expected angle of the events
A	Deployment area
$s \times s$	The size of the square sub-regions
c_1	Energy cost per bit of data transmission
c_2	Storage constraint

TABLE I: Summary of the symbols in TED

protocols for WSN, we assume the nodes will periodically send messages for link evaluation [16]. Therefore, the cost for maintenance is:

$$cost_{maintenance} = (\frac{A}{s^2})|N|c_1$$

Here, c_1 is constant that represents the relation between energy consumption and the size of the packets. In addition, the sensor node should also have storage constraint because the nodes simply might not able to store all the routes to every fusion point. The storage constraint is defined as:

$$(\frac{A}{s^2}) < c_2$$

Objective is to minimize:

$$cost_{all} = 2(\frac{T}{t}+1)cost_{forward}+cost_{maintenance}+Tcost_{detect}$$

All the constants are summarized in Table I. $cost_{all}$ may be obtained by nonlinear programming techniques such as generalized gradient search algorithm. In addition to square deployment, other deployment method may also be used and the only difference lies in Equation 2, 3 and 4.

VI. Evaluation

A. Analysis

In order to analyze the energy efficiency of TED without losing generality, we assume the sensor nodes are randomly deployed in a circular area with radius of R. We use distance to approximately measure the number of hops in order to calculate the message cost for event detection. As a reference to compare the energy cost, we use shortest path tree (SPT) algorithm where the events are collected at the sink because their definitions are not considered for the event detection.

We use a similar event model that has been introduced in Section V for analysis. Moreover, since the actual cost of TED will depend on event probabilities. We include such information in our model as well. Suppose we have two event types e_1 and e_2 which are the two sub-event types for a composite event e_3 (i.e. $e_1 r e_2 = e_3, r \in R$). The probability for e_1 and e_2 to occur is $P(e_1) = p_1$ and $P(e_2) = p_2$ respectively. The probability for e_3 to occur when both e_1 and e_2 have occurred is $P(e_3|e_1, e_2) = p_3$.

In TED, each node periodically broadcasts its routes to the fusion tables so that others can know how to reach the fusion points. Such cost is similar to many existing routing protocols in WSN such as [16] where each node periodically broadcasts its route metrics to the sink for the purpose of link quality evaluation. Therefore, in TED, we mainly consider three parts of energy consumption:

- The overhead for initial event forwarding: $cost_f$
- The overhead for the fusion points to send feedback: $cost_b$
- The energy cost for detecting the actual composite events: $cost_d$

Let the average distance between two random nodes in the square region be D and the average distance between a node and sink be d.

The cost for initial event forwarding will be: $cost_f = 2 \times D$. In order to avoid the extra cost for building up the overlay for each fusion point to communicate with individual sensor nodes, the fusion nodes simply flood the feedback in the network. Therefore, the cost for the fusion points to send feedback will be: $cost_b = D \times (|N| - 2)$. The cost for detecting each individual event is: $cost_d = D$. The total expected energy cost using TED over a time period T is:

$$\begin{aligned} & cost_{TED} \\ =& cost_f + cost_b + T(p_1 cost_d + p_2 cost_d + p_3 p_2 p_1 d) \\ =& D|N| + TD(p_1 + p_2) + Tdp_1 p_2 p_3 \end{aligned}$$

Here T is the $expire_n$ used in $table_e$. The total expected energy cost using SPT over a time period T is:

$$cost_{SPT} = d \times T(p_1 + p_2)$$

To see when TED will cost less than SPT, we have:

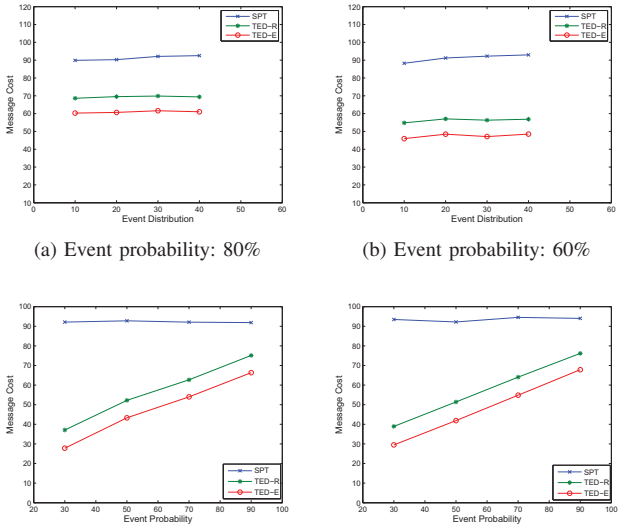
$$\begin{aligned} cost_{TED} &< cost_{SPT} \\ D|N| + TD(p_1 + p_2) + Tdp_1 p_2 p_3 &< dT(p_1 + p_2) \\ T((p_1 + p_2)(d - D) - dp_1 p_2 p_3)) &> |N| \end{aligned}$$

The inequality can be viewed as a trade-off between energy saved by TED and the overhead. Simply speaking, TED can save more energy when:

- Fusion points are closer to event source
- The probability of primitive event is high while the probability of the composite event is lower.
- Each time after the $table_e$ is constructed, it used for a relatively long period of T

B. Simulation

We use simulation to validate our analytical results. Our simulation is based on TOSSIM [17]. We have 127×127 sensor nodes placed on a square space. We use both even distribution and random distribution of the fusion points. Similar to our analysis, we divide the whole area into small squares and deploy equal number of fusion points for each square. Each sensor node is able to communicate with its nearby neighbors. To simulate the event detection, we first randomly generate event sources in the network. Then based on these event sources, we further generate more sub-events that have relations and let the sensor node detect the composite events. We study the performance of TED under different parameters such as event distribution, event probability, and deployment



(a) Event probability: 80%
(b) Event probability: 60%
(c) Event distance: 20
(d) Event distance: 40

Fig. 3: Simulation results

approach. In the figures, TED-R represents distributed TED with random fusion point deployment and TED-E represents distributed TED with even fusion point deployment.

Figure 3 shows all our simulation results. (a) and (b) show the performance of TED as the event distribution changes. The distribution changes are characterized by the change in the expected distance between the events. We study this type of events because they may be useful to certain applications where you want to detect co-related events that may be far away from each other. We also conducted the simulation based on different event probabilities. TED can outperform SPT by at least 10-20%. When the probability is lower, TED can further reduce the energy cost.

(c) and (d)are another set of simulation that shows the relation between cost and event distribution. In general, the energy cost of TED is linearly proportional to the event probability while the energy cost of SPT has little difference in regard to probability changes. This is because TED can efficiently filter a lot of events if their sub-events don't occur so these filtered events don't have to be delivered to the sink

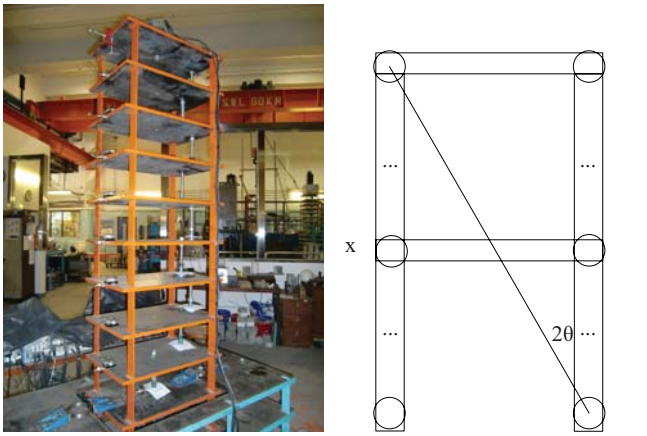
For all the simulations, we can find that the event probability is a major factor that affects the energy saving by TED. This makes TED particular useful in event-based systems where the primitive events occur very often while the composite events are more rare.

C. Experiments

We have implemented TED on MicaZ on top of PSWare. Similar to Section VI-A, we implemented a module that does opportunistic data aggregation based on the existing routing protocol provided by TinyOS for comparison.

We compare the performance using the following metrics:

- Message cost: this is obtained by setting up a counter inside the sensor node. The counter will be written into flash after the experiment so that we can retrieve it.
- Event detection delay: we measure the time between the subscription is disseminated and the event is notified.



(a) The testbed
(b) Parameters used in the experiments

Fig. 4: TED experiment setup

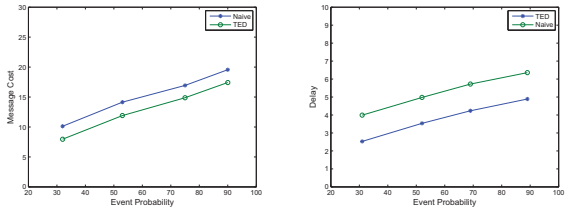
In our experiments, we consider the application for WSN-based Structural Health Monitoring (SHM) system. The objective of such system is to detect damages on structures such as buildings and bridges if they occur. Event detection is important in these applications because the SHM sensors will introduce high energy consumption during damage detection. It is therefore more desirable to wake them up only upon the occurrence of certain events [18].

Figure 4a shows our WSN-SHM testbed. In our experiment, we defined a scenario where the sensor nodes will start to collect the data when a certain vibration pattern is detected. The vibration is detected if the one sensor on the top and another one on the bottom of the model read the vibration data that satisfy certain criteria. The event definition is shown in Listing 2.

Listing 2: Event definition for SHM

```
1  Event Vibration {
2      data=System.Vibration;
3  } where {
4      data>THRESHOLD1
5  }
6  Event CompVibration {
7  } on {
8      Vibration e1 and
9      Vibration e2
10 } where {
11     e1.location=='top' &&
12     e2.location=='bottom' &&
13     e1.data-e2.data>THRESHOLD2
14 }
```

We manually generate events by hitting the model. Similar to our simulation, we implemented a naïve event detection



(a) Message cost (b) Delay

Fig. 5: Experimental results

method where the nodes simply use the existing routings provided by TinyOS [19] and detect events opportunistically. We implement two modules for both our TED and the opportunistic filtering where only the primitive events are filtered. In order to create multi-hop communication, we adjust the nodes communication range so that they can only communicate with nearby neighbors. For TED, we use tested both the centralized and the distributed versions and for the distributed version. For TED, we have the parameters set as shown in Figure 4b where x is the height of the model. The sensor nodes can communicate with other sensor nodes up to 2 floors away. According to the calculation, two fusion points were deployed in the model.

Figure 5 shows all our experimental results. In addition to energy efficiency, we also studied delay. We calculate the delay for TED as the duration between the time that naïve approach detects the event and the time that TED detects them. This is because naïve approach sends all the detected primitive events directly to the sink and the delay is only introduced by the multi-hop routing latency while both TED have additional delay in detecting the events. The experimental results are expected because TED slightly introduces more delay when doing the fusion points selection. The results on message cost is similar to our simulations.

VII. Conclusion

In this paper, we presented TED, a type-based composite event detection. We formulate the problem of composite event detection and found it to be NP-complete. Therefore, TED uses a distributed randomized algorithm. We have given some important mathematical equations on how to deploy fusion points for TED in an energy efficient way. We evaluated the performance of TED through analysis, simulation and experiments and compared it with naïve event detection where events are sent to the sink for detection. The results show TED can help to achieve high level energy efficiency without introducing too much delay.

Even though TED has good performance in detecting composite events, it may still have room for further improvement. For instance, current fusion point selection is done before the actual event detection. It might be possible if the fusion points are not only calculated before the event detection but also during the event detection process. We leave these as our future work.

Acknowledgment

This work is supported by Hong Kong RGC under CERG grant PolyU5102/08E and the Hong Kong Polytechnic University under Niche Area Project grant 1-BB6C.

References

[1] G. Liu, A. Mok, and E. Yang, "Composite events for network event correlation," in *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, May 1999, pp. 247–260.

[2] B. Lo, S. Thiemjarus, R. King, and G. Z. Yang, "Body sensor network - a wireless sensor platform for pervasive healthcare monitoring," in *The 3rd International Conference on Pervasive Computing*, May 2005, pp. 77–80.

[3] J. P. Lynch and K. J. Loh, "A summary review of wireless sensors and sensor networks for structural health monitoring," *Shock and Vibration Digest*, pp. 91–128, 2005.

[4] L. A. Klein, "Traffic parameter measurement technology evaluation," in *Proceedings of the IEEE-IEE Vehicle Navigation and Information Systems Conference*, August 1993, pp. 529–533.

[5] S. Gatziu and K. R. Dittrich, "Events in an active object-oriented database system," in *Proceedings of the first International Workshop on Rules in Database Systems*, 1993.

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114–131, June 2003.

[7] S. Lai, J. Cao, and Y. Zheng, "Psware: A publish / subscribe middleware supporting composite event in wireless sensor network," in *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, March 2009, pp. 1–6.

[8] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, March 2005.

[9] P. Bonnet, J. Gehrke, and P. Seshadri, "Towards sensor database systems," in *Proceedings of the Second International Conference on Mobile Data Management*, 2001, pp. 3–14.

[10] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, 2001, pp. 146–159.

[11] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, , and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, February 2003.

[12] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Ro, C. Ferraz, and J. Kelner, "Mires: a publish/subscribe middleware for sensor networks," *Personal and Ubiquitous Computing*, vol. 10, no. 1, pp. 37–44, December 2005.

[13] D. O'Keeffe, "Distributed complex event detection for pervasive computing," University of Cambridge/Computer Laboratory, Tech. Rep., 2010.

[14] D. Li, K. D. Wong, Y. H. Hu, and A. M. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, March 2002.

[15] B. Cârbunar, A. Grama, and J. Vitek, "Redundancy and coverage detection in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, pp. 94–128, February 2006.

[16] K. Srinivasan and P. Levis, "Rssi is under appreciated," in *Proceedings of the Third Workshop on Embedded Networked Sensors*, 2006.

[17] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, November 2003, pp. 126–137.

[18] S. Jang, H. Jo, S. Cho, K. Mechitov, J. Rice, S.-H. Sim, H.-J. Jung, C.-B. Yun, B. F. Spencer, and G. Agha, "Structural health monitoring of a cable-stayed bridge using smart sensor technology: Deployment and evaluation," *Smart Structures and Systems*, vol. 6, no. 5, pp. 439–460, July 2010.

[19] D. Gay, M. Welsh, P. Levis, E. Brewer, R. von Behren, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," in *Proceedings of Programming Language Design and Implementation*, 2003, pp. 1–11.