Adaptive pruning of Event Decision Trees for energy efficient collaboration in event-driven WSN

Steffen Ortmann, Michael Maaser, Peter Langendoerfer
IHP microelectronics
Im Technologiepark 25
15236 Frankfurt (Oder), Germany
Email: {ortmann|maaser|langendoerfer}@ihp-microelectronics.com

***Abstract*—Wireless Sensor Networks (WSN) are considered to be the key-enabler for low cost highly distributed applications in the area of homeland security, healthcare, environmental monitoring etc. A necessary prerequisite is reliable and efficient event detection. This paper introduces a novel approach for event configuration and in network processing, called Event Decision Trees (EDT). An EDT enables every node to self-divide event queries according to its resources. EDT autonomously adapt to the tasks assigned, even though it requires to organize collaboration between nodes to deliver expected results. The effort for maintain formal EDT is evaluated by analysis and simulations. Our results show that the proposed lease-based mechanism for maintaining producer/consumer pairs in an EDT outperforms even idealized Acknowledgment-based approaches.**

I. Introduction

Detection of real-world phenomena using wireless sensor networks (WSN) provides a basis for a variety of applications. Sensor networks can supply habitat and environment monitoring, context-awareness for personal services, smart homes, battlefield scenarios etc. [1], [2], [3], [4]. Traditionally sensor networks report their sensor readings to a global sink either continuously or if certain conditions are matched, e.g. a threshold is reached. Usually sinks are special nodes that provide more resources and take the final decision about sensed phenomena based on received sensor readings. Such data gathering applications exchange huge amounts of data, cause much traffic and large energy consumption and hence reduce the lifetime of the network.

Thus, only certain changes in sensor readings, called events, are transmitted. Events provide a suitable abstraction for sensor networks to state real-world phenomena [5]. We distinguish primitive and composite events. Primitive events usually describe the exceedance of a configured threshold by a single sensed value within the capabilities of a sensor node. Many applications demand detecting the simultaneous occurrence of several primitive events, particularly if identifying complex real-world phenomena is necessary. A combination of several primitive events is a composite event. For example, the occurrence of an event fire should be denoted as a combination of the primitive events *(temperature > 80°C) AND (smoke ≥ 1,1%)* instead of using the primitive events only. Further composite events based on different sensing capabilities indicating the same phenomena enhances the reliability of event detection [6].

Many approaches require the sensor nodes to provide all sensing capabilities needed for event detection. If that assumption cannot be granted the detection of composite events becomes much more difficult. In that case, sensor nodes must collaborate and share their sensing capabilities to continue with event detection. For reliable event detection we consider it a necessity to adapt the sensor nodes to different conditions automatically, i.e. heterogeneous distributed sensing capabilities, node mobility, changed network topologies, failed sensors or sensing units etc. We demand a suitable approach for composite event detection in sensor networks to consider the following design criteria:

Adaptivity - The capability of sensor nodes and applications to continue event detection when the context changes, sensors fail or nodes move. In particular, we focus on adapting collaboration between sensor nodes for automatic resource-oriented task distribution.

Autonomy - In addition to the autonomous nature of sensor nodes, every node in the network must be enabled to perform all necessary tasks for event detection. Thus, a fully decentralized approach is required since the usage of nodes acting as special entities like gateways [7] or fusion centers [8] results in potential Single Point of Failures (SPoF).

Transparency - Changing situation may trigger necessary adaptation and collaboration between sensor nodes to continue performing the configured task. Such changes must be transparent to the application, which is solely interested in events defined. Thus, changes must be provided automatically without concerning the specified event or the user.

Energy efficiency - Since collaboration simultaneously requires communication between sensor nodes, it increases the energy consumption and hence decreases the maximum reachable node lifetime. Thus, reducing the number of transmissions and the amount of exchanged data is of primary concern.

Applicability - Event detection schemes must be capable of dealing with heterogeneous nodes and network structures to support a maximum of multi-purpose applications and different hardware.

Convenience - We are interested in providing means that even allows the non-professional user to make use of sensor networks. Therefore a straightforward method to define events and configure sensor nodes without requiring knowledge about operating systems or programming languages for sensor net-

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

works is in demand.

We are aware of the fact that fulfilling all criteria up to a level of 100 percent is almost impossible and existing approaches usually tackle only a subset of those. We developed a concept for sensor network configuration considering all mentioned criteria. It allows to ignore low-level details like node resources, network structures, node availability etc., and enables the programmer to work on a high global abstraction level only, i.e. the event itself including related constraints. If an event is defined, our system autonomously and transparently assures that this event is configured and monitored, even though it requires to organize collaboration between nodes to deliver the results.

The contribution of this paper is an intuitive XML-based event definition language that simplifies event configuration to a level that is even suitable for non-professionals, who are usually short on experience of programming languages and sensor networks. Based on that, we introduce a novel fully decentralized mechanism to autonomously set up distributed event detection called Event Decision Tree (EDT) and a cost efficient means to maintain such EDT. An EDT enables every node to self-divide event queries according to its own resources and self-adapt to the tasks assigned. Simultaneously, the EDT provides the interface for efficient collaborative event detection between neighboring nodes using a lease-based publish/subscribe approach. EDT can be efficiently constructed on every device by using a tiny Generating Finite State Machine (GFSM) requiring ten states only. The simulations clearly show that our concept works well and the applied collaboration scheme outperforms even idealized ACK-based approaches.

The next section presents related work and discusses respective advantages and drawbacks. Section 3 gives an overview of our system architecture. Section 4 introduces our event specification language for sensor network configuration. It features hardware independent description elements that allow correlating heterogeneous sensing capabilities to specify composite events for sensor networks. Section 5 presents our distributed event detection scheme based on adaptive pruning of EDT's. Section 6 examines our collaboration scheme, compares its efficiency to an idealized ACK-based approach and points out simulation results. Finally we examine our contributions and conclude with a summary and an outlook on future work.

II. Related Work

Event based sensor networks become very popular in the research community. Many approaches that tackle the special problems of composite event detection in sensor networks have been published recently. We head for two directions, the event specification and decentralized collaborative event detection, and discuss both fields. Below we present published approaches and point out advantages and drawbacks before introducing our composite event detection scheme.

Many higher abstractions for sensor network programming and configuration are already available. One of the most famous is TinyDB [9] extending SQL to support in network data queries. It provides a good abstraction layer to specify data collection in database-query style but still works on the node level. Thus, the application programmer still requires knowledge about node resources, locations, etc. Further, TinyDB requires a complex query interpreter on the nodes and uses a centralized topology with at least one coordinating node to interpret received data. That creates a SPoF and requires a huge amount of traffic to continuously collect raw sensor reading. Macro programming languages such as STOP [10] allow to create data queries from a global viewpoint without considering details of single nodes. Based on agents, which migrate through the network and collect data according to the query, STOP provides a more comfortable data collection. But it requires a complex runtime environment and virtual machines on every node. Even here, collected data must be analyzed by central nodes (SPoF). Furthermore both approaches require to use scripting or programming languages, which is not feasible for non scientific deployment. We demand a straightforward configuration concept that is tailored to the user. Thus, the user only needs to define what he is interested in, i.e. the event and event-related constraints. In other words, the user need not to take care of hardware, software and node deployment because every node is enabled to self-configure to the event defined.

In addition, we focus on enhancing autonomous collaboration for reliable composite event detection, even in case of missing resources, mobile nodes or failures in sensors and connectivity. We present some approaches for collaborative event detection, point out their basics and evaluate these approaches against the introduced criteria in Table I. Vu et al. introduce a composite event detection scheme for sensor networks composed of different nodes with varying sensing capabilities [7]. They split the problem of composite event detection among different nodes into sets of atomic events, which are similar to threshold values. Atomic events are reported to a special gateway node (SPoF) that finally checks whether a composite event has emerged. This approach provides configurable levels of fault tolerance by selecting an appropriate k for k-watching sets of sensors while taking care of the energy consumption and the event notification time but requires an expensive setup phase. Kumar et al. present a similar collaboration scheme [11]. They create event based trees for each composite event, which contain all participating sensor nodes. Connection between all nodes is established using a content-based publish/subscribe method from children to parent nodes. Even here, the root node obtains all sensed measurements and comes to a final decision about the monitored event (SPoF). Kamiya et al. describe a composite event detection system [12] where heterogeneity is given by using several heterogeneous sensor networks. They apply a P2P network of special sensor gateways each accessing and managing a certain sensor network. To define composite events over one or more networks, the sensor gateways provide an XML-event description parser that registers and subscribes the needed atomic events at the corresponding gateway nodes. Atomic events are determined by the gateway nodes only and hence, the underlying sensor networks have to report

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

	Vu [7]	Kumar [11]	Kamiya [12]	CoDED [13]
Adaptivity	(+)	(+)	(0)	(++)
Autonomy	(0)	(-)	(- -)	(+)
Transparency	(0)	(0)	(+)	(0)
Energy eff.	(0)	(- -)	(- -)	(-)
Applicability	(+)	(++)	(++)	(+)
Convenience	(0)	(0)	(+)	(-)
Validation: (++) very good; (+) good; (0) regular; (-) bad; (- -) very bad				

TABLE I
COMPARISON OF RELATED WORK ACCORDING TO DESIGN CRITERIA.

the raw sensed data to the centralized nodes which is not energy efficient. Further, all mentioned approaches shift the final decision to a certain node and are vulnerable if that nodes are faulty or completely fail. The CoDED platform [13] presents an architecture for context-dependent composite event detection in sensor networks. In order to save energy resources, events are monitored in certain monitoring context only. The context description is defined by a propositional logic, which evaluates to true as long as a specified context is given. Combinations of primitive events may form global (and maybe distributed) composite events, which are observed by a composite event detection engine. That engine seems to adapt automatically to current network situations but the general question of how to distribute and process composite events on several devices is left open. This approach was not implemented so far.

All presented approaches lack means in meeting every requirement listed in Section I. As seen in Table I, most provided is adaptivity and applicability whereas energy efficiency, autonomy and convenience are marginally taken into account or are completely missing. To the best of our knowledge there exists no approach that associates all introduced requirements. Nonetheless, some ideas of these approaches inspired us to combine them in a new suitable event detection scheme that tackles all design criteria mentioned.

III. ARCHITECTURE

To give an overview of our approach, Figure 1 displays the architecture of our event detection system. Our system consists of two major components, the event packet generator on the server side and the event configuration environment on every sensor node. Please note, we provide a detailed description of all necessary steps for event configuration in the next sections.

At the server side, the application programmer uses our XML event specification language to define events she is interested in. Every XML event specification passes three automatic steps before being distributed in the sensor network. First, an XML parser generates the respective XML tree representation. Thereafter the human readable XML elements of that tree are adapted to the targeted sensor system, i.e. conversion of values for sensing, renaming of identifiers and functions etc. Finally the adapted XML-tree is converted and compressed into a minimal deployable event definition, called event packet. These are distributed in the sensor network for initial event configuration, updates and deletion as well.

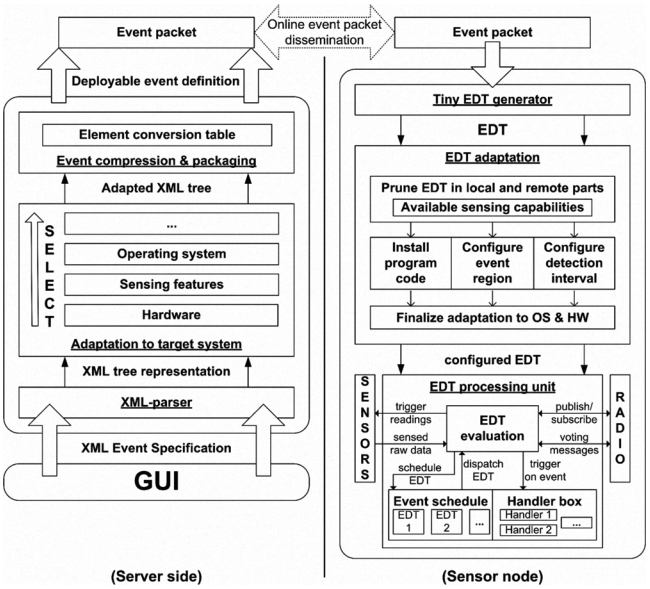

Fig. 1. Architecture of our event detection system. It consists of two major components, the event packet generator on the server side and the event configuration environment on every sensor node. Defined events are disseminated in the network as event packets.

On the sensor nodes, every incoming packet is processed to generate the respective event representation as EDT. According to the sensing features and resources provided at the node, the EDT is split into local and remote parts. Local parts can be evaluated by the node itself, whereas remote parts have to be requested from external sources, e.g. neighboring nodes. After further adaptations and configurations of event related constraints, the final EDT is integrated to the EDT processing unit. The EDT processing unit autonomously collects required sensor readings, frequently evaluates the EDT with respect to the configured detection interval, manages necessary collaboration with other nodes and triggers associated handlers in case of positive event evaluation. Please note, the EDT processing unit is enabled to administrate and process several EDT at the same time.

IV. SPECIFYING COMPOSITE EVENT DETECTION

In this section we introduce our event specification language. The major goal is to provide means, which allow to define event detection in a straightforward and intuitive way. Our approach is an enabling technology for easy event definition and in-network processing. It allows the programmer to ignore all low-level details and to concentrate on a high global abstraction level, that is the event itself and its related constraints. By that it can be used by human users but might as well become part of a “middleware” like approach.

Our language enables to specify primitive and composite events. It defines precise operating thresholds for heterogeneous sensing capabilities and enables to combine those by logic operations to specify composite events. Configurable execution intervals and appropriate event handlers can be assigned to events. Further a region of event can optionally

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

be defined for each event. Our approach also supports multi-event evaluation on sensor nodes. This is a basic feature to ensure self-adaptability in case of changing situations. In other words it is a prerequisite to ensure flexibility. Thus, sensor nodes must be enabled to store and process several event specifications simultaneously. We integrated an updating mechanism analogous to code update means as those provided e.g. by Contiki [14]. Such a feature allows easy reconfiguration or recalibration of already deployed sensor networks.

A. Event specification structure

Our language uses an XML-styled structure[1] and specifies an event within a tag <EVENT>. An event specification consists of four main elements. The <SENSORDATA> element defines the required sensing capabilities for primitive or composite events respectively. Each event specification contains an <EXECUTION> element that states the frequency of event detection. Appropriate processes, which are triggered upon positive event evaluation are listed in the <CONSEQUENCE> element. The optional <DIMENSION> element defines the expansion of the *region of a certain event*. The event region is given by a configurable radius around the initiating node and contains all devices, allowed to participate in a distributed event detection. IF this element is omitted, the 1-hop neighborhood is considered the default event region.

For configuring several event specifications simultaneously we embedded further attributes in every <EVENT> element. An event-"id" assigns a globally unique identifier to events, which enables to associate requests and updates to a certain event. The "version" number identifies different versions of the same event specification. It reduces maintenance and online reprogramming complexity. Incoming event specifications with higher version numbers substitute all older versions of the targeted event specification, i.e. those with lower version numbers. Due to scarce memory resources all older versions are deleted and only the current version is stored. Our language supports multi-event evaluation by providing a means to assign priority levels to every event by using the "priority" attribute. Consider a sensor network that is used for gathering temperature measurements for climate control but is used in parallel to detect forest fires. In such a setting the detection of forest fire would have the higher priority because it is a safety-critical event.

The "lease" parameter and the "reliableMode" attribute allow the programmer to customize the adaptiveness and efficiency of the communication scheme used for collaboration between neighboring nodes. A "lease" defines the frequency of adaptation between neighboring nodes. In other words, it specifies the time lag between two adaptation phases where usual event processing takes place. Self-adaptation is required to overcome the problems of varying context, fluctuating environment and node mobility but consequently requires an communication overhead. Short lease intervals (small lease factor) provide a high adaptation rate whereas long lease intervals can significantly reduce the number of messages and energy consumption. The "reliableMode" attribute allows the programmer to choose between a higher reliability in data exchange or a reduced energy consumption. Enabling the "reliableMode" instructs to explicitly acknowledge every data exchange that is correlated to a certain event. Thus the reliable mode consequently requires a communication overhead but enhances the reliability of detection. Thus, safety-critical events should make use of the reliable mode, whereas simple data collecting scenarios could omit the required overhead in favor of less energy consumption. It is quite obvious that configurations of both parameters strongly depend on the application as well as the application context. Further descriptions of this parameter is given in Subsection VI-A.

[1]We are aware of the fact that XML is not well suited for use on sensor nodes. We pre-parse the specifications into appropriate packets, introduced in an extra section, before deploying them on sensor nodes.

B. Combining heterogeneous capabilities

Heterogeneous sensing capabilities offer great advantages to enhance the reliability of event detection in sensor networks. Combinations of different sensor features enable more precise and complex event detection capabilities. Almost all sensor network applications define threshold values for certain measurements, called primitive events, and fire an event if current sensor readings match or exceed these values.

Our approach provides means to combine several sensing capabilities and their respective threshold values. The <SENSORDATA> element enables to list sensing capabilities and to configure corresponding primitive events for their measurements. Primitive events can be defined as exact values or quantified as single-bounded scopes by using relational elements, such as <EQUAL>, <LESS>, <GREATER> etc. Relational elements define respective relations between two elements, which are variables and/or constants. A variable identifies a sensing capability and is defined by the <VARIABLE> element. Thus, the value of a variable is given at run-time by sensor readings. In contrast to that, the <CONSTANT> element defines a constant value and a respective unit, which is used as threshold. The "unit" attribute allows for assigning different units, e.g. time and distance units like "seconds"or "meters". Converting the specified constants with respect to the hard- and software used on the sensor nodes, e.g. converting seconds to milliseconds if necessary, is task of the language interpreter and is not of concern for the user.

To support composite and even heterogeneous event specifications, primitive events can be composed by logic operations. Logic operations are specified by own tags called <AND>, <OR> and <NOT>. Additionally it supports defining 2-bounded scopes for certain sensing capability by defining several primitive events for the same capability. For example, defining an event for measuring a temperature between 20 and 25 would result in a combination of two primitive events, i.e. $(temp \geq 20)\ AND\ (temp \leq 25)$.

C. Execution intervals and associated handlers

Energy consumption is an essential issue when designing WSN based applications. Therefore we are providing means

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

that help to adjust the energy consumption of the event evaluation process. Sensor nodes provide different modes of operation that result in significant different energy consumption. Active modes like data processing or data transmission are draining the energy resources much more than passive modes such as sleeping [15]. Thus, active periods must be kept as short as possible in favour of passive periods in order to reduce energy consumption to a very minimum. On the other hand, extensive passive periods may reduce the accuracy and reliability of event detection. Real-world phenomena are usually subject to different temporal resolution, which must be considered for event specification as well. For example, the acoustic wave of an explosion can only be detected within a few milliseconds and hence require a short sensing interval.

When a node may switch to a power saving mode depends highly on the application running. Thus, an event specification contains an <EXECUTION> element that allows configuring application-oriented execution intervals. It enables the definition of separate time-oriented execution intervals for each event. These time intervals can be quantified by acceptable periods or exact time slots that must be adhered. Intervals are set using a relational attribute and a constant as threshold. We additionally intend to integrate means that allow configuring resource-oriented execution intervals, e.g. scaling the event evaluation interval due to expiring energy resources. Finally the <CONSEQUENCE> element links procedures to an event. The procedures, called event handlers, have to be executed in case of a positive event evaluation. Every event handler is listed by a <TRIGGERHANDLER> element, which contains the name of the event handler. Specifying several event handlers for a single event is allowed and all of them are executed in the sequence as listed, if that event occurs.

D. Example

To illustrate our event specification language we discuss an example of a composite event specification in this subsection. In this example a sensor network is used for fire detection. Beside other criteria, a fire can be detected by monitoring the ambient temperature or the emission of smoke. Widely used fire detectors set off a fire alarm if monitored smoke emissions exceed a given threshold. Also changes in temperature can be analyzed to detect a fire [16]. In spite of using well-engineered sensing devices both methods are still vulnerable to false alarms. Appropriate combinations of several different detection methods - in our example temperature and smoke - allow to enhance the reliability of detection and to decrease the false alarm probability at the same time.

Listing 1 displays an event specification that can be used in fire detection scenarios. It defines a *fire* event as a combination of temperature and smoke readings. Therefore an ambient temperature limit of at least 353 Kelvin and a smoke emission limit of 1.1 percent are defined as primitive events. Hence, if and only if the measurements for both parameters exceed their threshold values the measuring sensor node has detected the composite event *fire*. We assume a radius of 2.5 meter around the sensor node as a suitable region for distributed detection

```
<EVENT id="fire" version="1" priority="high"
       lease="6" reliableMode="yes">
 <SENSORDATA>
  <AND>
   <GREATER>
    <VARIABLE> temperature </VARIABLE>
    <CONSTANT unit="Kelvin"> 353 </CONSTANT>
   </GREATER>
   <GREATEROREQUAL>
    <VARIABLE> smoke </VARIABLE>
    <CONSTANT unit="percent"> 1.1 </CONSTANT>
   </GREATEROREQUAL>
  </AND>
 </SENSORDATA>
 <DIMENSION>
  <REGION-CIRCLE relation="LessOrEqualTo">
   <CONSTANT unit="meters">2.5</CONSTANT>
  </REGION-CIRCLE>
 </DIMENSION>
 <CONSEQUENCE>
  <TRIGGERHANDLER> sendalert </TRIGGERHANDLER>
 </CONSEQUENCE>
 <EXECUTION>
  <TIMEINTERVAL relation="EqualTo">
      <CONSTANT unit="seconds">10</CONSTANT>
  </TIMEINTERVAL>
 </EXECUTION>
</EVENT>
```

Listing 1. Example of an event specification for fire detection scenarios in sensor networks

of the *fire* event. Hence, the dimension element defines that region by specifying a maximum distance of 2.5 meters. In case of having evaluated the composite event *fire* to be positive the sensor node triggers the "sendalert" event handler.

V. Deployment on Sensor Nodes

XML-styled event specifications provide flexible and easy to use configuration means for developing sensor network applications, even for non-experts in the field. We consider this a necessary issue for widely using sensor networks beyond the scope of research, e.g. by medical employees adapting them for customized patient monitoring. Nevertheless, strict energy and memory constraints in WSN demand further processing of event specifications for node configuration. This section presents all steps required for configuring sensor nodes according to event specifications. This includes separating complex events into less complex ones based on the sensing facilities of individual sensor nodes as well as means to detect nodes, which can provide the missing information to evaluate the complex events. Please note, every process described from now is autonomously done by our system and remains fully transparent to the application programmer.

A. Pre-Parsing and Packaging

Since XML is oversized for direct use on sensor nodes, event specifications are pre-processed before in-network deployment. Therefore event specifications have to pass our XML-parser. Similar to the event specification, an event packet

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

fire1h6y	10	<=D$2.5	sendalert	&>temperature,353.0>=smoke,1.1

Fig. 2. Pre-parsed event packet of the fire detection example provided at Listing 1.

consecutively describes all basic elements. Keeping a given order allows to describe all elements by their content only. Event packets are applied for initial event configuration as well as for event updates, i.e. reconfiguration or deletion.

To minimize the calculation effort on the sensor nodes, our XML-parser converts event specifications into appropriate packets for deployment. At this point, the parser allows to create hardware-specific event packets of every universally valid event specification. It is quite obvious, that general event descriptions cannot be uniformly transferred to every sensor platform due to different hardware used, e.g. equal sensing capabilities measured by different sensors with varying physical units. To overcome these problems, the XML-parser converts given values in the specification into required ones for the target sensor platform. For example, temperature values given in centigrade are converted to Kelvin or time data is converted from minutes to milliseconds if necessary. To keep the event specification quite simple and intuitive for the user, conversions are fully-transparent to the user and automatically done by the parser.

Event specifications are first parsed into the respective XML-tree representation. Constants and variables are adjusted to meet the requirements of the target sensor platform, as mentioned. It is important to omit all redundant information to keep an event packet as small as possible to save resources during packet transmission and processing, particularly to reduce the energy consumption of the sensor nodes. Using short notations for compound elements and a strict packet layout can reduce the size of event packets by more than 90 percent compared to the event specification. Starting with the event header, all specified elements are successively transformed into minimized descriptions. The event header consists of the event id, the version number as well as the priority, the lease and reliableMode parameters. In the shortened form, all elements are associated to one string. The tags of the XML-language elements are represented by symbols. The sensor data and the dimension element are converted to a minimized prefix (or polish) notation of the respective XML subtree. The prefix notation places operators to the left of their operands. Since the arity of the logic and relational operators is fixed, which is here one for the NOT and two for all other elements, the result is a syntax without brackets. This syntax can still be parsed without ambiguity. Listed event handlers and the execution interval are also added to the packet. Event handlers are consecutively listed and separated by comma. The execution interval is determined as number specifying the time between triggering two successive event evaluation processes. Depending on the target sensor platform it is usually specified in seconds or milliseconds.

Event packets are transmitted as Byte-Streams to the sensor nodes whereby each element is separated by a colon. That reduces the size of the fire detection example by a factor of 9 from 560 Bytes + whitespace down to 63 Bytes, provided as packet displayed in Figure 2. Using Boolean bit arrays representing the sensing capabilities can decrease the packet size further. Since the variety of sensing features is large and differs from platform to platform, we refrained from specifying the bit arrays but considered that as an extension in the design of the language parser.

B. Generating Event Decision Trees (EDT)

Event packets must be parsed at the sensor nodes to generate evaluable event configurations. Therefore the sensor node establishes Event Decision Trees (EDT) representing the event based on the sensor data element of the event packet. The EDT enables every node to self-divide event queries according to its resources and to execute the complete event evaluation process. Nodes cannot only be used as data source for sensing and distributing raw data. In fact, every node is qualified to analyze and process its sensor readings and hence, to come to a final decision about the occurrence of events, which is "yes" or "no". Further, the EDT is a fully distributed concept that does not require special nodes for data collection. We consider this mandatory to prevent from single point of failures, which are naturally arising if only one or a few nodes are enabled to execute the complete process.

To parse the prefix notation of the sensor data element into a congruent representation as EDT, we developed a tiny GFSM that can parse the complete language using only ten states. That enables to implement the parser on every available sensor platform. Relational elements provide the basis of the EDT. A relational element is described as a tree structure, where the parent node constitutes the respective relation to its children nodes. According to the specification of primitive events, children of relational nodes can either identify a sensing capability or a constant value (threshold). If composite events are specified, the primitive events are respective subtrees. Logic nodes, representing the logic combinations of several primitive events, are generated as parent nodes on top of all relational ones. In the fire detection example, the root node of the EDT represents a Boolean AND between both primitive events regarding smoke and temperature. The equivalent EDT is depicted in Figure 3. For further processing the tree nodes are pre-order numbered during their creation from the event packet. That assures the same initial tree-node labeling on every device in the network, which is necessary for efficient exchange of event information later.

C. Evaluation of EDT

EDT nodes can be automatically evaluated in a bottom-up manner starting from the leave nodes in order to determine a Boolean value at the root node, i.e. the final event evaluation result. EDT evaluation is triggered by internal event-related timing constraints (sensing interval) or by requests from other devices. All nodes of the EDT representing sensing capabilities are assigned with actual sensor readings. In our example these

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

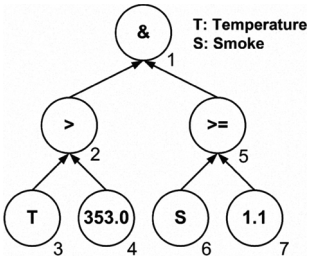

Fig. 3. Pre-ordered Event Decision Tree of the fire detection example.

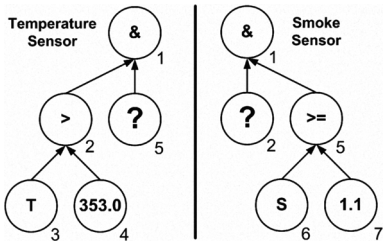

Fig. 4. Pruned Event Decision Trees for smoke and temperature sensors monitoring the introduced fire detection event. "Undecidable" nodes are marked by a "?".

are the nodes "T" for temperature and "S" for smoke readings. Afterwards the EDT is bottom-up evaluated by comparing the children of each node according to the function defined at the parent node. As a result, Boolean values are assigned to relational and logic nodes. If the value of the EDT root node was evaluated to TRUE, i.e. the event was detected, all specified event handlers are triggered for further processing.

D. Pruning of distributed EDT

Until now, it was assumed that sensor nodes possess all sensing capabilities to evaluate the complete EDT on its own. Additionally we focus on enabling sensor nodes to evaluate the EDT even if they do not provide all or no sensing capabilities needed. That could be either by design or by failed sensing units. Hence certain branches or subtrees and the corresponding nodes of the EDT cannot be evaluated. In that case, sensor nodes need to collaborate to exchange information about sensor readings. The exchange of sensed raw data, which is done by most approaches, is very inefficient from two point of views. First, permanently exchanging sensor reading leads to a huge number of transmissions and hence, consumes much energy and reduces network performance. Second, transmitting raw sensor data includes to use large data packages, depending on the number of readings and their accuracy, i.e. the size of every value usually varies from 1 to 4 bytes. Since one of our main goals is to remain very energy efficient, we concentrate on minimizing the number of transmissions and the amount of data to be exchanged. Instead of exchanging raw sensor readings we propose to process sensor readings first and finally submit 1 bit only, which states whether the threshold for the sensor reading is exceeded or not. Please note, there already exist some approaches that share information in a "yes" or "no" style, but these can only state the occurrence of the complete event. We are focusing on efficiently sharing information about both, complete and partial events.

In our case, we only need to transfer the Boolean value of a certain EDT node. Missing node values may be delivered by neighboring nodes that share the specified *region of event*. To prepare these data exchanges, every sensor node has to determine which node information is missing at the locally generated EDT. The following algorithm prunes the established EDT until it contains the minimum required EDT for local event processing:

1) Mark each leave as pruned that represents a sensing capability which is not supported.
2) Search all nodes that possess at least one marked child excluding the root node[2].
   2.1 Mark current node as pruned, if
      a All children are marked as pruned or
      b The unmarked child represents a constant (threshold).
3) Repeat step 2 until no new nodes are marked. After that, all undecidable subtrees are marked.
4) Prune all marked nodes except for the root nodes of the marked subtrees.
5) Declare all left marked nodes as "undecidable".

After pruning, the EDT may contain nodes which are marked as "undecidable". Respective Boolean values must be obtained by suitable sources, e.g. neighboring nodes in the *region of event*. Lets assume to use separate temperature and smoke sensing nodes for the introduced fire detection example. Hence the EDT depicted at Figure 3 must be pruned with respect to the available sensing capabilities. The temperature node cuts the branch containing the smoke readings and the smoke node cuts the branch containing the temperature readings respectively. That results in two different EDT at the sensing devices, each containing one node marked as "undecidable". Thus, the temperature sensing device requires information about tree node number 5 whereas the smoke sensing unit requires information about tree node number 2. The resulting EDT are displayed in Figure 4. We excluded the EDT root node from the pruning algorithm to enable nodes that possess no suitable sensing capability for event detection, to serve as "bridge". Such nodes are of interest if they are located between two or more nodes required for event detection that cannot communicate directly. Hence, all nodes deliver their part of event information to the "bridge" node, which is finally enabled to decide about the occurrence of that event. After having identified the "undecidable" parts for event detection on each sensor node, a suitable collaboration scheme is required to efficiently share necessary information.

VI. Collaborative Exchange of Event Information

Wireless sensor nodes should communicate if and only if it is absolutely necessary to save energy resources. A suitable

[2]Since an EDT is a binary tree, every node possesses at most two children. Hence either one or both child nodes are marked as pruned in that case.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

collaboration mechanism in sensor networks must self-adapt to changing network situations and consider application requirements. We propose to apply an adaptive and easy-to-scale publish/subscribe scheme to maintain exchange of EDT node values. To ensure a certain level of reliability for collaborative event detection, some basic principles have to be discussed from different points of view of subscribers and publishers. How do subscribers know whether some other node received the subscription, accepted it or is still providing publications. On the other side, the publisher must know whether there is still a subscriber requiring event information.

Most problems could be solved by using a simple Acknowledgment (ACK) for every transmission to inform the sender about the success. Unfortunately that produces a huge amount of traffic and is therefore inefficient for sensor networks. Furthermore we intend to reduce the traffic by submitting changes of node states only to achieve longer time intervals without any transmission. That case cannot be considered by ACK schemes, which require to communicate at each detection interval to renew the subscription interest. Due to changing network conditions, publications and subscriptions should either be removable or be valid for certain time periods only. The latter is much more suitable for sensor networks where changes mostly happen unforeseen and hinder appropriate responses. Thus, we apply a lease procedure to the publish/subscribe scheme that requires significantly less transmissions and enables fine-tuned event-defined lease intervals.

1) Subscribe a data interest: Missing values of "undecidable" EDT nodes must be obtained by suitable other sources as mentioned. Thus, the sensor node broadcasts a data interest (subscription) into the network. A subscription contains the label of the EDT node and the event identifier. If the event specification defines a region of event, the subscription must also contain the location of the subscribing sensor node. On receiving a subscription, the sensor node compares own and received location data to determine whether both nodes share a region of event. Only if that holds true or if the request contains no location data, i.e. the default event region determined by sending range, the received subscription is of interest. The receiving sensor node searches its own respective EDT to determine whether it can provide the requested information. In that case, the requested EDT node is marked with a "publish" flag and the sensor node answers the request by providing the current state of the requested EDT node. In all other cases the node discards the received subscription without further processing. Subscriptions can further consist of many concurrent data interests in case of requiring information about several "undecidable"EDT nodes of one or more events. That significantly reduces processing and communication effort required for packaging, addressing, transmission etc.

2) Publishing EDT node information: On event evaluation the current state of each EDT node is determined. Results at nodes marked with the publish flag are also important for other devices in the network and hence may be published. To save resources these evaluation results are not transmitted permanently. Only two cases demand transmitting the current node state, that are first-time subscriptions and state changes. If a device accepts a received subscription for the first time, it must answer with the current node state to provide an initial value for the subscriber. Since a node state is of Boolean type, only state changes must be submitted to update the current node state later. If node states change rarely, the number of required publications is significantly reduced. Even in the worst case, i.e. the node state changes at each evaluation, this scheme requires the same overhead as usual methods where values are transmitted continuously at every evaluation period.

3) Minimizing the amount of exchanged data: Since every bit to be transmitted is expensive in view of energy consumption, the amount of exchanged data must also be minimized. Our EDT allow to share event information efficiently by using a few bytes only. In contrast to existing approaches that need to share raw sensor data, here only the Boolean value of a certain EDT node is of interest. Thus, a data transmission must only contain the event identifier, the number of the respective EDT node and the current Boolean value assigned to that node. Please note, establishing and prefix-numbering the complete EDT before pruning assures that the EDT at each sensor node possess the same numbering.

We use an efficient labeling scheme that allows to describe the node of interest and the assigned value with one byte only. That byte consists of 1 bit representing the Boolean value and 7 bits representing the address (number) of the EDT node. That way, 128 different nodes in one EDT can be numbered. If an EDT contains more than 128 nodes, an extra byte for labeling is used. In addition, the event identifier must be submitted given that a sensor node is enabled to configure several events and respective EDT concurrently. To simplify matters, here we used readable event identifier (fire) for our examples. If the event identifier is chosen to be a unique number less than 256, all necessary information can be transmitted by two bytes only.

A. Applying lease periods for publications and subscriptions

Using a publish/subscribe scheme is rather simple if reliable architectures and fixed network structures are provided, but sensor networks are subject to unpredictable behavior triggered by changes in context, connectivity, working mode etc. That especially holds true if mobility of nodes is provided. Thus, the publish/subscribe scheme must adapt frequently to reach a certain level of reliability in event detection. Certainly, the overhead needed for adaptation must be kept as small as possible but still allow for balancing the adaptiveness with respect to the application provided.

We propose to subscribe for a certain lease period only. A lease-based subscription specifies a certain time interval determining the validity period of subscriptions that is the time during which associated publications have to be sent. On receiving a subscription the node determines the expiration date of publications. The expiration date e depends on the event evaluation interval i and the application-defined lease factor k, which are both given by the event specification. e is

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

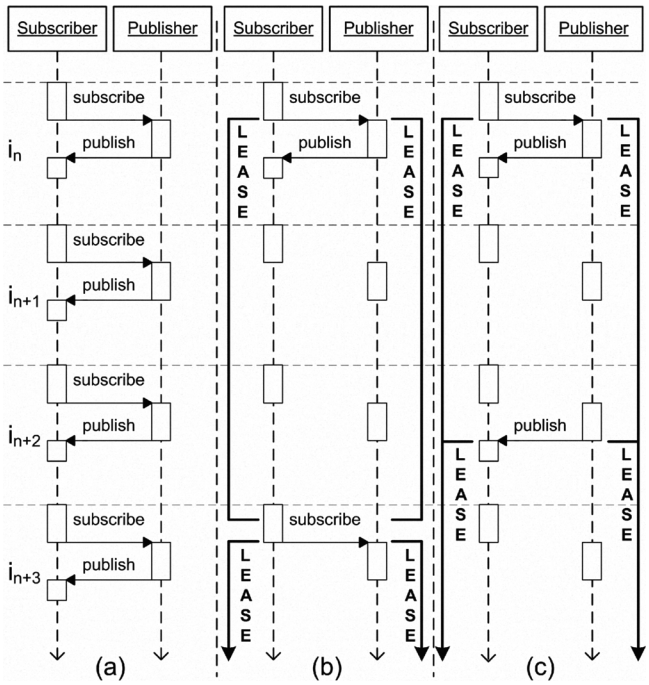

Fig. 5. Sequence of information exchange between a single subscriber and a single publisher over time of 4 event evaluation intervals i_n to i_{n+3}. (a) displays the performance of the ACK-based variant. (b) and (c) illustrate the performance and the lease allocation in case of non-event (b) and event (c). Both apply a lease factor of 3, i.e. every subscription is valid for 3 event evaluation intervals.

calculated using equation (1).

$$e = i * k;\ k \geq 1 \tag{1}$$

The lease factor k allows the user to adapt the lease period to the monitored event as well as to the expected conditions in sensor networks. For example, sensor networks which are subject to permanent changing situations or node mobility require a high adaptiveness by using short lease intervals. In contrast to that, sensor networks deployed at rather fixed network structures could make use of larger lease intervals to save energy and extend the overall network lifetime. The expiration date is assigned to the corresponding EDT node together with the "publish" flag. After initial publishing the current node state, any further change is published as long as the "publish"flag is set. Consequently, the flag is automatically removed from the EDT node when the expiration date is reached, i.e. the lease has expired.

To save more energy, we distinguish new and renewed leases to save the initial respond of the publisher too, since it is not needed if no change occurred. If earlier agreed leases are to be renewed only, the publisher does not respond with the initial node value but extends the lease period and continues providing state changes until the newly assigned expiration date is reached. In addition, publisher and subscriber renew the lease period automatically upon notification of a state change. Figure 5 displays sequence charts of both lease extension cases as well as the ACK-based scheme for comparison.

i	event evaluation interval
k	lease factor ($k \geq 1$)
N_s	number of subscriber
N_p	number of publisher
T	sequence of detection intervals
p_t	event probability ($p_t \leq 1$)
n_s	number of subscriber for a single publisher

TABLE II
PARAMETERS AND TERMINOLOGY USED FOR EFFICIENCY ESTIMATION.

B. Efficiency (Comparison)

In order to prove the efficiency of using a lease-based distribution of event information, we compare the cost of the introduced lease procedure to a communication scheme based on Acknowledgments. The required traffic is analyzed considering different points of view of a single subscriber and a single publisher and finally estimated for the entire sensor network. Additionally, we show that there exists a break-even point for the lease procedure for almost all scenarios in contrast to the ACK-based scheme. Even the worst-case scenario, which exists in theory only, results in marginal overhead for the lease procedure.

Many published projects have proven that links in WSN are unreliable. According to this, the number of originally required messages in the network increases by a certain amount. We are aware of that fact, but since we are comparing two different communication schemes under same network conditions, both would assign such traffic increase. Thus, to simplify the estimation, we do not consider unreliable links for direct complexity comparison and calculate the essential required traffic only. Consequently, the approach that performs better in the idealized network condition case, does so with unreliable links too. The simplest way to assure reliable collaboration among sensor nodes is to use ACK-based communication. To provide a proof of efficiency, we made a best case scenario assumption of the ACK-based scheme. That is, it requires exactly two messages per event detection interval. We neither regard that several publisher may answer to the same subscription nor that several subscriber may acknowledge the same published value simultaneously. Obviously, both cases would increase the required traffic. Explicitly confirming each subscription or publication establishes a form of bilateral relationship. It informs the subscriber that there really exists a suitable publisher and that one gets a feedback that there is still some subscriber requiring information. Disappearing subscribers or publishers can be recognized immediately. The drawback is that every data exchange requires two transmissions per evaluation interval to reconfirm the relationship. According to this, a subscriber either sends one subscription or one acknowledgment for received publications per detection interval. Publishing nodes respectively publish data as subscribed or answer new subscription requests by one message per interval i. Expression (2) determines the required messages within a consecutive sequence of detection intervals T for distributing event information using an ACK-based publish/subscribe

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

scheme. Table II lists the parameters used for traffic estimation here and later.

$$T * (N_s + N_p);\ T = n * i, n \in \mathbb{N} \tag{2}$$

A lease-based approach eases the strong relation of ACK-based communication in favor to less overhead. Due to the fact that the validity of a subscription extends automatically, the lease procedure assures that publishing event information is performed if and only if it is necessary. That adapts publications to subscriber requirements periodically and saves a lot of energy at the publisher side. Since each subscription assigns an expiration date a subscriber needs to renew a subscription when the expiration date is reached. Hence a subscriber sends a subscription message not at every interval but after every e. A publisher sends a one-time message to new subscribers to provide the initial node value. Afterwards a publisher notifies its subscriber(s) if and only if the Boolean node value changes. We consider a probability of changes p_t. If the last subscriber disappears during an active lease, there exists a chance of unnecessary publishing data but only for a maximum of $(k-(Tmodk)-1)$ intervals. Hence, the required traffic for a lease-based publish/subscribe is calculated using expression (3).

$$N_s\frac{T}{k}+N_p((T-1)p_t+n_s+(k-(Tmodk)-1)p_t);\ p_t \leq 1 \tag{3}$$

To compare both approaches we analyze both schemes from the points of view of subscribers and publishers. With regard to subscribers the lease-based approach clearly outperforms the ACK-based variant, see equation (4). Even considering the worst-case, i.e. a lease extends after every interval $k = 1$, results in equal cost of communication.

$$TN_s \geq \frac{T}{k}N_s \ \Leftrightarrow\ T \geq \frac{T}{k} \ \Leftrightarrow\ 1 \leq k \tag{4}$$

From the view of publishers the cost analysis and comparison is more complex. That requires evidencing the validity of equation (5), which is equivalent to (6). To summarize, there exists a T that satisfies equation (6), unless $p_t = 1$. In other words, after a certain number of intervals even here the lease-based procedure performs better than the ACK-based. Only in case of publishing data at every interval because of permanently toggling events $p_t = 1$, the lease procedure requires a small overhead. With growing number of intervals, that overhead becomes nearly irrelevant. Moreover, the probability of $p_t = 1$ is not existent in real applications and therefore becomes negligible.

$$TN_p \geq N_p((T-1)p_t + n_s + (k-(Tmodk)-1)p_t) \tag{5}$$

$$\frac{T-n_s}{T+k-(Tmodk)-2} \geq p_t \tag{6}$$

Finally we carry out a break even point analysis to show that the lease-based procedure always outperforms the ACK-based approach after a certain number of intervals. Therefore the validity of equation (7) has to be proven. The break-even point for the lease-based scheme can be easily determined by transformation to T, see formula (8). As it is easy to see, equation (8) is solvable except that $k = 1$ and $p_t = 1$ at the same time. Whereas choosing $k = 1$ is possible but not reasonable, the case of $p_t = 1$ is rather unlikely as mentioned.

$$T(N_s+N_p) \geq N_s\frac{T}{k}+N_p((T-1)p_t+n_s+(k-(Tmodk))p_t) \tag{7}$$

$$T \geq \frac{N_p(n_s+p_t(k-(Tmodk)))}{N_s(\frac{k-1}{k})+N_p(1-p_t)} \tag{8}$$

C. Simulation results

To give a proof of concept as well as to test our approach under different conditions we implemented the EDT on the discrete event simulator OMNeT++ [17] with an extension for wireless sensor networks, called Castalia [18]. The introduced fire detection scenario was simulated to measure the number of messages needed for collaborative event detection.

We used the following parameters to test and compare the performance with different lease factors k. The network consists of 100 sensor nodes – 50 nodes measuring temperature and 50 nodes measuring smoke. Since the detection of the event *fire* requires smoke and temperature measurements, all nodes play the roles of subscriber and publisher at the same time. The nodes are uniformly distributed in a field of 20x20 meters. According to the event specification, the detection of the *fire* event ranges over a radius of 2.5 meters around the related sensor node. A deterministic event generator simulated the occurrence of events, i.e. appropriate smoke and temperature readings, with an event probability of ten percent. We applied two different network structures. The simulation results for the first scenario, a regular uniform sensor grid, are displayed in Figure 6. Please remember, one interval i represents ten seconds in lifetime. Accordingly, $k = 6$ means that renewal of the leases and adaptation of the event detection is done every minute.

The second test bed used different networks of random uniform distributed sensor nodes to simulate more realistic applications. The simulation results for the second test bed vary only marginally from the first test bed. Therefore we waived to display the diagram for the second test bed. Comparing the simulation results to the estimation shows that the number of required messages closely meets the estimated traffic. Due to changing network structure and varying node placement, the second test bed required marginal less average traffic. Altogether, no simulation run exceeded the estimated traffic. Depending on the event probability and the lease factor, our lease-based publish/subscribe approach significantly reduced the number of required messages compared to the ACK-based variant. For the given scenario, a lease factor of $k = 6$ reduces the number of messages by a factor of 8, whereas $k = 60$ reduces the required traffic by a factor of 18.

VII. Contributions

Higher abstraction for convenient event definition. Our event specification language hides low level details of WSN to focus on pure event definition. The XML-styled language

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856

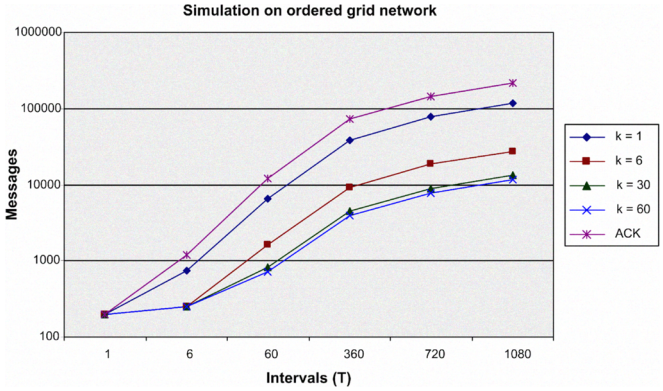

Fig. 6. Simulation results of applying the lease-based publish/subscribe approach to the fire detection scenario. Depending on the chosen k, the lease-based approach requires significantly less overhead in communication.

enables to correlate required sensing features and assign event-related constraints. All necessary processing and adaptation for deployment remains fully transparent to the user.

A novel decentralized mechanism to autonomously set up event detection and in-network processing on sensor nodes, called Event Decision Tree (EDT). An EDT enables every node to self-divide event queries according to its own resources into local and remote parts by pruning. Using EDT every node in the network can execute the complete evaluation process without a single point of failure, even in case of missing sensing features or failed units.

A cost efficient means to maintain such EDT. The EDT provides the interface for efficient collaborative event detection between neighboring nodes using an lease-based publish/subscribe approach. It outperforms even idealized ACK-based approaches and reduces the required traffic and energy consumption. Appropriate on-node processing of sensed data allows to efficiently share event information by a few bytes only. Additionally it allows to configure the adaptiveness as well as the energy consumption to the application.

VIII. Summary&Outlook

In this paper we identified missing features of composite event detection in wireless sensor networks, particularly in energy consumption, reliability and complexity, as a relevant shortcoming. To remedy this shortcoming we combined a flexible event definition language with a self-adapting event detection scheme. Specified events are deployed on the sensor nodes as Event Decision Trees (EDT). An EDT self-adapts to varying network and node conditions by automatic pruning. That enables reliable application, even in case of missing sensing facilities or failures of nodes. A lease-based publish/-subscribe scheme manages necessary collaboration between sensor nodes. Our approach was successfully implemented and tested. The simulation results show that the proposed lease-based mechanism for maintaining EDT outperforms even idealized ACK-based approaches.

By design, our approach also provides mobility of sensor nodes. We are currently investigating whether self-adapting event detection in mobile sensor networks can be supported. Further interesting areas of application are infrastructure supported networks and MANETs.

References

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *WSNA '02: Proc. of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2002, pp. 88–97.

[2] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, 31 Jan.–2 Feb. 2005, pp. 108–120.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[4] M. Aboelaze and F. Aloul, "Current and future trends in sensor networks: a survey," in *Proc. Second IFIP International Conference on Wireless and Optical Communications Networks WOCN 2005*, 2005, pp. 551–555.

[5] K. Romer and F. Mattern, "Event-based systems for detecting real-world states with sensor networks: a critical analysis," in *Proc. the 2004 Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2004, pp. 389–395.

[6] K.-P. Shih, S.-S. Wang, P.-H. Yang, and C.-C. Chang, "Collect: Collaborative event detection and tracking in wireless heterogeneous sensor networks," in *Proc. 11th IEEE Symposium on Computers and Communications ISCC '06*, 2006, pp. 935–940.

[7] C. Vu, R. Beyah, and Y. Li, "Composite event detection in wireless sensor networks," in *Proc. IEEE International Performance, Computing, and Communications Conference IPCCC 2007*, 2007, pp. 264–271.

[8] T.-Y. Wang, Y. Han, P. Varshney, and P.-N. Chen, "Distributed fault-tolerant classification in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 724–734, 2005.

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

[10] H. Wada, P. Boonma, and J. Suzuki, "A spacetime oriented macro-programming paradigm for push-pull hybrid sensor networking," in *Proc. 16th International Conference on Computer Communications and Networks ICCCN 2007*, 2007, pp. 868–875.

[11] A. V. U. P. Kumar, A. M. R. V, and D. Janakiram, "Distributed collaboration for event detection in wireless sensor networks," in *MPAC '05: Proc. of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM, 2005, pp. 1–8.

[12] H. Kamiya, H. Mineno, N. Ishikawa, T. Osano, and T. Mizuno, "Composite event detection in heterogeneous sensor networks," *IEEE/IPSJ International Symposium on Applications and the Internet*, vol. 0, pp. 413–416, 2008.

[13] S. Schwiderski-Grosche, "Context-dependent event detection in sensor networks," in *2nd Intl. Conf. on Distributed Event-Based Systems (DEBS'08)*, Rome, Italy, July 2008.

[14] A. Dunkels, O. Schmidt, T. Voigt, and M. Ali, "Protothreads: Simplifying event-driven programming of memory-constrained embedded systems," in *Proc. of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, November 2006.

[15] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *SASN '06: Proc. of the fourth ACM workshop on Security of ad hoc and sensor networks*. New York, NY, USA: ACM, 2006, pp. 169–176.

[16] D. Herbert, V. Sundaram, Y.-H. Lu, S. Bagchi, and Z. Li, "Adaptive correctness monitoring for wireless sensor networks using hierarchical distributed run-time invariant checking," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 2, no. 3, p. 8, 2007.

[17] A. Varga, "Omnet++," *"Software Tools for Networking", IEEE Network Interactive*, vol. 16, no. 4, 2002.

[18] H. N. Pham, D. Pediaditakis, and A. Boulis, "From simulation to real deployments in wsn and back," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007.*, June 2007, pp. 1 – 6.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6856
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6856