Knowl Inf Syst (2017) 50:231–264
DOI 10.1007/s10115-016-0942-z

REGULAR PAPER

Rule-based inference and decomposition for distributed in-network processing in wireless sensor networks

Ozgur Sanli[1] · Ibrahim Korpeoglu[2] · Adnan Yazici[1]

Received: 26 February 2014 / Revised: 21 February 2016 / Accepted: 23 March 2016 /
Published online: 7 April 2016

Abstract Wireless sensor networks are application specific and necessitate the development of specific network and information processing architectures that can meet the requirements of the applications involved. A common type of application for wireless sensor networks is the event-driven reactive application, which requires reactive actions to be taken in response to events. In such applications, the interest is in the higher-level information described by complex event patterns, not in the raw sensory data of individual nodes. Although the central processing of information produces the most accurate results, it is not an energy-efficient method because it requires a continuous flow of raw sensor readings over the network. As communication operations are the most expensive in terms of energy usage, the distributed processing of information is indispensable for viable deployments of applications in wireless sensor networks. This method not only helps in reducing the total amount of packets transmitted in the network and the total energy consumed by the sensor nodes, but also produces scalable and fault-tolerant networks. For this purpose, we present two schemes that distribute information processing to appropriate nodes in the network. These schemes use reactive rules, which express relations between event patterns and actions, in order to capture reactive behavior. We also share the results of the performance of our algorithms and the simulations based on our approach that show the success of our methods in decreasing network traffic while still realizing the desired functionality.

Keywords In-network processing · Event-driven applications · Rule-based information processing · Wireless sensor networks

✉ Ibrahim Korpeoglu
korpe@cs.bilkent.edu.tr

[1] Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

[2] Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey

1 Introduction

With recent advances in micro-electromechanical systems (MEMS) that enable the development of tiny sensor nodes, it is now possible to establish a wireless sensor network (WSN) that can be deployed in a region to monitor it and sense the physical phenomena occurring there. A typical wireless sensor network is composed of a large number of tiny sensor nodes that have sensing and radio communication capabilities along with limited processing and storage resources. Sensor nodes generally operate on power supplied by irreplaceable batteries, and thus, energy efficiency is one of the most important objectives in the design of WSNs and related techniques. When the energy consumption in a sensor node is considered, it is found that sensing and processing operations use far less energy compared to communication operations. As described in [33], transmitting 1 kb of data 100 m costs around three joules in a noise-free environment. Nevertheless, the same amount of energy is consumed by a general purpose processor with 100 MIPS/W capability when it executes around three million instructions. These facts suggest that reducing traffic volume is of major concern for the energy-efficient operation of WSNs.

In-network processing is a distributed information-processing approach that can be used in a WSN to reduce its traffic volume. With in-network processing, all or a part of the data generated by sensor nodes is processed by the sensor nodes themselves, rather than all data being processed at the sink. As a result, refined higher-level information is transported inside the network instead of a huge amount of raw sensor data.

When in-network processing is employed, sensor readings can be processed near the sensed area and redundant or unnecessary data can be filtered out. In this way, the amount of traffic transported in the network decreases, which improves energy efficiency and prolongs the lifetime of individual sensor nodes and the entire sensor network. Furthermore, processing data inside the sensor network result in a more fault-tolerant and scalable solution. If a centralized approach is used for information processing, a sink node takes all the responsibility of processing and becomes a single point of failure and may also become a bottleneck point. The energy problem cannot simply be solved by adding more sensor nodes, since this will incur more deployment and management costs and may cause more sensor data and control traffic to be transported.

An important challenge associated with WSNs is that different kinds of applications have different sets of requirements, which makes it impossible to design a network that will satisfy the needs of all possible application scenarios. For example, some applications are deployed to gain insight on an unknown physical phenomenon. Such research-oriented applications require all sensor readings to be collected at a center and recorded for offline analysis. On the other hand, forest-fire detection or healthcare monitoring applications are expected to provide notification of emergencies. As these examples show, a sensor network and its related algorithms may need to be designed considering the specific application in mind.

Sensor network applications can be classified as demand-driven and event-driven applications [39]. In a demand-driven application model, sensor nodes send their data when a central node requests it. In an event-driven application model, on the other hand, the processing and reactions take place only after the occurrence of something important to the application, and there is no need for a continuous flow of sensor readings or periodic query requests. In order to efficiently implement such event-driven applications, we need to establish mechanisms inside the network that can detect events and produce high-level information from the raw data, decide whether the information is of interest to the application user at the center, and if so send the relevant information to the sink node or other appropriate nodes in the network.

There are increasing number of application scenarios for wireless sensor networks that require event detection. In sensor networks, tracking, surveillance or detection applications typically depend on the occurrence and the detection of composite events. Habitat monitoring [35], patient monitoring [13], volcano monitoring [21], coal mine monitoring [22], forest-fire detection [24] and structural health monitoring [23] are examples of such applications. They require a large-scale event system that needs to detect composite events, which provide a high-level abstraction of raw data, efficiently and quickly. Cougar [38] and TinyDB [28] are some of the popular data-processing schemes that employ in-network processing in sensor networks. Cougar and TinyDB view a sensor network as a large distributed database, and instead of collecting data at the sink and processing there, queries are pushed into the network and processed inside the network in a distributed manner. Cougar employs threshold-based event detection, while TinyDB composes specified attribute thresholds. Main objective of these schemes is query processing. One of the first schemes on event-driven sensor networks was directed diffusion [16], which is a query-driven data aggregation method for WSNs. A number of event-driven routing protocols have been proposed as well [5,26,29]. TEEN [29] is an event-driven routing protocol, where routing decisions are based on data exceeding some threshold values. TTDD [26] is based on a grid topology and requires sensor nodes to be stationary and location-aware. RR [5] performs event flooding which means it is only suitable for environments where the number of events is very small. In [36], an in-network aggregation scheme is described. In order to reduce the number of messages exchanged, a routing tree is constructed that maximizes the number of overlapping nodes which in turn increases the aggregation rate. Such protocols consider networking issues ahead of the data processing and often limit data processing to simple operations like thresholding or use of basic aggregation operators.

Several schemes have been proposed that employ composite event detectors inside a hierarchical network similar to our study [20,32]. In [32], composite event expressions are decomposed into sub-expressions to be detected inside the network. Similarly, the study in [20] proposes a complex event-processing framework, based on a three-layer network, transforming events into sub-events which is detectable by ordinary nodes. Neither of them discuss about how sub-expressions or sub-events are generated. Furthermore, the former scheme is not specifically about sensor networks and therefore does not take constraints of sensor networks, whereas the latter scheme merely discusses framework components at conceptual level, without enough details.

In this paper, we propose an in-network processing method for WSNs that distributes information processing to sensor nodes, while still realizing the required application functionality and logic that is achieved by centralized processing. Our method adapts rule-based inferencing as the information fusion method. A set of event–condition–action (ECA) rules in a rule-base expresses the application logic; however, instead of storing and using the rule-base at a central node, we introduce a decomposition algorithm, RBDA, which decomposes the rule-base into several sub-rule-bases so that rule processing can be distributed into appropriate nodes in sensor networks. In this study, a hierarchical network is considered, where there are different node types with different sensing capabilities and responsibilities. For each type of node, a new sub-rule-base is created from the original rule-base. At the end, the original central rule-base is decomposed into multiple sub-rule-bases that are used to construct information-processing engines to be distributed into the sensor network.

Although RBDA shows a simple, easy-to-grasp and rational way of distributing information processing, it possibly suffers from exponential space complexity. Therefore, we introduce a new way of representing rules and adapt our algorithm so that it can handle these new rules. We call our new algorithm as v-RBDA. We show the results of experiments which confirm that v-RBDA removes the deficiencies of RBDA.

Our methods are particularly useful for large-scale sensor and actor networks that run event-driven and real-time applications. In such applications, transmitting all raw sensor measurements without considering whether they are related to an event of interest causes too much energy consumption. Additionally, for real-time applications that require an immediate response, the delay between the occurrence of an event and the respective decision and the accompanying action should be small and bounded. In large networks, this delay may be quite large if processing occurs only at a central location.

Main contributions of our study are as follows: (1) we present a hierarchical WSN architecture where nodes are classified and given roles in a hierarchical manner depending on what they can process. (2) We provide a generalized form of the antecedents of rules to be used in rule-based inferencing for event-driven reactive sensor network applications. (3) We propose a rule set decomposition algorithm that generates sets of rules from an original rule-base. (4) In order to improve space complexity of our initial rule set decomposition algorithm, we introduce a new way of representing rules and adapt our decomposition algorithm so that it can handle the new representation. (5) We describe an application scenario where we apply our algorithm that could benefit from our approach and show how our approach can be applied. (6) We conduct extensive simulation experiments and present the results showing the effectiveness of our methods in decreasing network traffic while still realizing the desired functionality.

The rest of the paper is organized as follows: In Sect. 2, we present the related work. In Sect. 3, we first give preliminary information and then describe a mechanism that distributes rule processing into the sensor network. While the mechanism is simple and easy to grasp, it introduces too much space and processing overhead. Therefore, in Sect. 4, we introduce an extension of that mechanism that uses a different way of representing rules and performs better. In Sect. 5, we discuss an application scenario that might benefit from our approach. In Sect. 6, we discuss about the experiments regarding the performance of our algorithm, and in Sect. 7, we present the results of simulations that show the improvements. In Sect. 8, we elaborate more on our methods and their applications, and finally in Sect. 9, we conclude the paper.

2 Related work

Lifetime of sensor networks depends heavily on lifetime of individual sensor nodes, and a typical sensor network consists of battery-powered sensor nodes [2]. Therefore, energy efficiency is an important parameter for almost all of sensor networks. On the other hand, a single energy-saving scheme cannot be employed for different sensor networks as the requirements differ from application to application. Proposed power management strategies include reducing the number of sensors that are active, and reducing the number of communication operations or the amount of data transmitted. In order to reduce the active sensors at a given time, a subset of sensors representative of the whole sensor network is selected based on spatiotemporal correlations [3,19]. Contextual information may also be used to control sensor operations to reduce sensing or communication operations [8]. Regarding reducing networking operations, energy-saving techniques can be grouped into three [15]: the first approach is MAC-layer sleep scheduling in which sensor nodes are put into sleeping mode for a specific amount of time so that energy is saved due to avoiding idle listening state during sleeping period [6,25,27]. This approach is suitable for an environment where events rarely happen and late reaction may be tolerable. The second approach is to reduce the amount of information transformed by using some techniques like compression or coding [11,30]. This is especially useful if all of the sensor readings need to be stored in a central location. The final approach is to process information inside the network, which is called in-network processing, in order to reduce total number of network transmissions as the network operations are the costly operations in terms of energy consumption. Although central processing shall generate more precise results, in-network processing approach may be the viable solution for applications that are event driven and place energy efficiency and/or timely response ahead of precision.

Cougar [38] and TinyDB [28] are some of the popular data-processing schemes that employ in-network processing in sensor networks. Cougar and TinyDB view a sensor network as a large distributed database, and instead of collecting data at the sink and processing there, queries are pushed into the network and processed inside the network in a distributed manner. These systems, however, are query based and are therefore demand driven, hence, not very suitable for the event-driven applications on which we focus in this paper. In directed diffusion [16], interests, which in that case are attribute-value pairs, are propagated into the network by the sink node. The sensor nodes that have matching data to the interests send their data toward the sink node along the gradients setup, while interests were being propagated. The data are then aggregated on the way back to the sink. Similar to previous approaches, this approach is also demand driven and not very suitable for applications that are event driven and require continuous monitoring of a region to detect events. Furthermore, attribute-value pairs are not always expressive enough to describe what is of interest to the application's users and unnecessary packets might still be transported in the network.

Some of the approaches that resemble to our approach are as follows: in [31], a rule-based distributed fuzzy logic reasoning engine is described. The reasoning engine employs simple if–then rules for the decision process and it uses fuzzy logic to fuse individual sensor readings and neighbor observations to get more accurate results. Although if–then rules are used for information processing, the main motivation of the study in [31] is to improve the reliability of the decisions, and the paper focuses on and discusses only information processing at a single node. However, our method concerns distributed information processing and inference in an entire WSN, and in our approach the sensor nodes in a network cooperate to reduce network traffic and energy consumption.

Event-driven applications need to identify composite event patterns. The study in composite event detection was first conducted in active database field. In [12], a mechanism suitable to model the semantics of composite events and the implementation of an event detector for active databases are described. In [32], a composite event-processing framework that works on top of a range of publish/subscribe systems is proposed. Distributed composite event detectors are installed at various locations in the network according to the requirements of the application, such as latency, reliability or bandwidth usage. Although this work is similar to our study in that composite event expressions are decomposed into sub-expressions, similar to what we do in our rule-base decomposition scheme, their proposition is not specifically about WSNs and therefore does not take the constraints and features of WSNs into account. Furthermore, they do not provide a specific method for decomposition. In our work, however, we give a specific method and algorithm about how a rule-base, expressing the application logic of a wireless sensor network, can be decomposed into sub-rule-bases while still realizing original functionality. Additionally, we consider a hierarchical network architecture and we describe precisely how we can classify WSN nodes based on what they can do and sense, and therefore where the information-processing engines and sub-rule-bases can be placed.

In [17], complex events with temporal and spatial constraints and correlations are detected in a sensor network by employing a hierarchical approach. Events are classified as mote-level, group-level and base-level events. At the mote level, individual readings from sensor nodes are processed and the outputs of this processing are fed into the inputs of the next level. The results of group-level processing are similarly used by the base level, where the final decision about a composite event that the application is interested in is made. Although a hierarchy of event detectors is used in this paper, the paper does not discuss how the detectors are created nor on what criteria they are based. On the other hand, we show how we classify different hierarchies from an information-processing perspective.

In [1], a proactive and distributed mechanism is proposed to detect a set of interrelated events, also called contexts, in a sensor network. For this purpose, event notifications are delivered to special nodes that are connected through an overlay network. These nodes make partial context decisions and forward their decisions to the next node in the overlay. Our work differs from [1] by expressing logical relations as conjunctions of disjunctions of premises. Additionally, Ahn and Kim [1] uses an overlay and processes the decisions in overlay nodes. Our approach, however, does not use an overlay and is based on a hierarchal structure of a WSN. When an overlay is used, it has to be maintained and nodes need to know their overlay neighbors and how to route data to them. In our approach, however, nodes only need to know how to reach the nearest nodes in the next level, which simplifies the routing strategy.

In [10], a rule-based flexible programming environment for wireless sensor networks is described. In order to deal with resource limitations of sensor networks, they propose a rule filtering and merging algorithm. Rules that cannot be processed by a sensor node are filtered out from the rule-base of it. Similarly, rules for the same event but from different sources are merged.

3 Rule decomposition for distributed rule processing

There are obvious benefits to centralized data processing. For example, a global view of data may bring more accurate results and only the processing resources of the central node need to be considered. Nevertheless, as we have discussed before, this paradigm is generally not suitable for wireless sensor networks and therefore data need to be processed in a distributed manner.

In this section, we first give some preliminary information, and then we introduce our first rule-decomposition algorithm, RBDA, which generates a set of sub-rules enabling distributed processing of central rules.

3.1 Preliminaries

3.1.1 Event-condition-action rules

In our approach, application logic is expressed by a set of ECA rules that represent a set of statements. The execution of these statements depends on the occurrence of specific event patterns and the satisfaction of the constraints on the events that constitute those event patterns. The statements express the actions to be taken and the conclusions to be drawn.

Employing ECA rules instead of embedding the logic into the application code has several advantages [39]:

1. Rules can be stored outside the application in a rule-base, which improves the modularity, maintainability and extensibility of the applications.
2. Rules have a high-level declarative syntax, so they can easily be analyzed and optimized.
3. Rules provide a generic mechanism to express reactive behavior, contrary to the application code, which is typically specialized to a particular type of reactive scenario.

In a procedural paradigm, how the application reacts to the events should be coded by the application developers. However, by employing a declarative, rule-based paradigm, domain experts may directly determine the type of reaction. Application developers need therefore only develop a general purpose rule engine to process rules. Such separation of duties is favorable, because in this way each person does what he is most proficient at.

ECA rules are in the form of: IF <*event pattern*>, PROVIDED THAT <*set of constraints*>, THEN <*set of actions*>. The first part of a rule contains the event pattern that needs to be detected in order for that rule to fire. However, the detection of the event pattern does not guarantee the triggering of the action statements in the third part of the rule. The context in which the event pattern has been detected is of profound importance for intelligently determining the implications of the detected events. The conditions and constraints that help in evaluating the context are listed in the second part of the rule.

3.1.2 Events

The IF-part, also called the antecedent, of a rule contains one or more events, i.e., a set of events. We define an event as any change in the state of the data being monitored where the change has significance for the application. The arrival of a network packet or a value exceeding some threshold might be typical examples of WSN events.

A single event on its own is not generally enough for drawing conclusions or taking actions. In order to capture a real and complicated application scenario, we need to use more complex event patterns, formed by the logical composition of simple events using the operators of an event algebra. Conjunction, disjunction, sequence and negation are four core composition operators used in event algebras that have been introduced in the literature [9]. Other operators have also been proposed, such as the concurrency operator, which requires two events to occur in parallel, or the iteration/counting operator, which specifies a specific number of occurrences of some event. Most of these extra operators are the results of efforts that try to combine the temporal and logical relationships of the events.

In our study, we take only *conjunction, disjunction* and *negation* as the logical operators. We do not use the sequence operator, which is listed as a core operator above, as a logical composition operator, because sequence can be defined as conjunction with a before relation between component events.

3.1.3 Node classification

Sensor nodes in a WSN can be classified into various types depending on the heterogeneity of the hardware or the logical roles that the sensor nodes possess. Physical capabilities and responsibilities of nodes might differ: two nodes may sense different stimuli or one of the nodes may contain more powerful processor or enhanced battery resource compared to the other. Hardware heterogeneity explicitly indicates that nodes with differing hardware cannot assume the same role. However, even in WSNs that are homogeneous in terms of hardware, different roles might exist. Roles in such networks are assigned to nodes according to what logical responsibilities they will assume. We can think of several logical roles that are related to the network and information-processing architecture employed. In a clustered network, a node may take the role of a source that initiates the network traffic, a relay that simply forwards other nodes' packets, a cluster head that acts as a gateway for communication with other nodes outside of the cluster, a sink that is the ultimate destination, etc. In terms of information processing, a node might be a data-disseminating source, an aggregator, a data fusion node or assume another similar role.

In the context of our study, a role emphasizes what a node can process. There might be various data types involved in data processing. For example, sensory data, such as temperature, pressure and acceleration or data that are generated as a result of aggregation or fusion, are data types that might be present in a WSN. Let T be the set of all different data types appearing in a WSN and $P(T)$ denote the power set of T. Then the set of roles can be defined as $R = \{r \mid \exists r \in P(T)\}$ with the cardinality m such that:

1. $T = \bigcup_{i \to 1}^{m} (r_i)$
2. Given $r_i \in R$, $r_j \in R$, $1 \le i \le m$, $1 \le j \le m$, and $i \neq j$, one of the following needs to be satisfied:

   (a) r_i and r_j are mutually disjoint,
   (b) $r_i \subset r_j$ or
   (c) $r_i \supset r_j$

The first condition denotes that every data type is assigned to a role such that it will be processed by one type of a node in the network. The second condition states that two roles do not intersect unless one of them is the proper subset of the other. 2.a ensures that a physical phenomenon is sensed by only one type of a node. In a hierarchical sensor network, we assume that nodes that are in the higher levels have the ability to see the data of the lower levels, because sensor nodes send their data to the nodes on the upper levels. Conditions 2.b and 2.c represent this assumption.

Classification of the nodes is the process of determining every possible r_i that encloses the set of data types and pertains to the above constraints and conditions. Roles are determined in a bottom-up fashion: starting from the available data types, each data type is assigned to one of the appropriate roles. Roles are precomputed, and assignment of them to the actual nodes might be static or dynamic. However, role assignment is not the subject of this paper. We use roles as inputs in the rule-decomposition algorithm, which is described in the following subsection.

3.2 RBDA: rule-base decomposition algorithm

The rule-base decomposition algorithm we propose first creates the sub-rule-base for the type of nodes that are classified at lowest hierarchy. Unlike the other types of nodes, such nodes' input set does not have any subset which is the input set of other types of nodes. If such a subset existed, then the these nodes would not be on the lowest level; higher layers have more inputs, which include inputs from lower layers.

Next, the rule-base will be created for the nodes with the property such that subsets of those nodes' input set can only be the input set of a node from a lower level, not from the higher levels. This iterative process is repeated until a sub-rule-base is created for each different type of a node. The data are processed in the lowest possible level, and it is not relayed into the upper layers. This bottom-up approach in creating the sub-rule-bases supports the idea that "data should be processed as close to its source as possible".

The antecedent of a rule is in the form of conjunctions or disjunctions of predicates. A predicate is a binary function that takes a variable number of arguments and returns a value of true or false. Antecedents need to be put into the conjunctive normal form in order to decompose rules. The conjunctive normal form is the conjunction of disjunctive clauses:

$$(P_{11} \mid P_{12} \mid \cdots \mid P_{1k}) \mathbin{\&} \cdots \mathbin{\&} (P_{i1} \mid P_{i2} \mid \cdots \mid P_{in})$$

Each P_{ij} is a predicate. For any node in the sensor network, the predicates of a rule to be executed in this node can be classified into two: predicates that might be processed by the node and predicates that might not. Using this information, we can rewrite the antecedent in the following generalized format:

$$P \mathbin{\&} (Q_1 \mid Q'_1) \mathbin{\&} (Q_2 \mid Q'_2) \mathbin{\&} \cdots \mathbin{\&} (Q_n \mid Q'_n) \mathbin{\&} R,$$

where P is the conjunction of disjunctive clauses that only contain predicates that can be evaluated given the node's input set; i.e., all the necessary input for the evaluation of P is available at the node. $Q_1, Q_2, \ldots, Q_n$ are the disjunctions of the predicates that the node can evaluate, whereas, $Q'_1, Q'_2, \ldots, Q'_n$ are the disjunctions of the predicates that it cannot. Note that the expression Q' is not the negation or inverse of Q. The disjunction of these two produces the disjunctive clause $(Q \mid Q')$ in the rule. Finally, R is the conjunction of the disjunctive clauses that only contain predicates that this node cannot evaluate.

RBDA takes the central rule-base and the input set of the type of node that we are intending to create a sub-rule-base for as the inputs. Let q be the set $\{Q_1, Q_2, \ldots, Q_n\}$ and q' be the set $\{Q'_1, Q'_2, \ldots, Q'_n\}$, which are constructed using the input set of the node. Furthermore, let $\mathbb{P}$ be the powerset of q. For a member M of the powerset $\mathbb{P}$, if M and q are not identical,

Algorithm 1 RBDA

```
1: INPUT: $IS$=input set, $RB$=original rule-base
2: OUTPUT: $RB$=original rule-base (modified), $NRB$=new sub-rule-base
3: **for all** $Rule$ in $RB$ **do**
4:     Set $P, R, q, q'$
5:     $\mathbb{P} \leftarrow powerset(q)$
6:     **for all** $M$ in $\mathbb{P}$ **do**
7:         **if** $M$ equals to $q$ **then**
8:             **if** ($R$ is NULL & ($P$ is not NULL | $q$ is not empty)) **then**
9:                 Add [($P$ & $conjunction\_of\_elements\_of\_M$) $\rightarrow O$] into $SRB$
10:            **else if** $q$ is empty **then**
11:                **if** ($P$ is NULL) **then**
12:                    Add [$R \rightarrow O$] into $RB$
13:                **else**
14:                    Add [$P \rightarrow O_k$] into $NRB$
15:                    Add [($O_k$ & $R$) $\rightarrow O$] into $RB$
16:                **end if**
17:            **else**
18:                Add [($P$ & $conjunction\_of\_elements\_of\_M$) $\rightarrow O_k$] into $SRB$
19:                Add [($O_k$ & $R$) $\rightarrow O$] into $RB$
20:            **end if**
21:        **else**
22:            Add [($P$ & $conjunction\_of\_elements\_of\_M$) $\rightarrow O_k$] into $SRB$
23:            $M' \leftarrow \{Q'_i \mid Q_i \notin M\}$
24:            Add [($O_k$ & $conjunction\_of\_elements\_of\_M'$ & $R$) $\rightarrow O$] into $RB$
25:        **end if**
26:    **end for**
27:    Remove $Rule$ from $RB$
28: **end for**
```

then a rule having the following antecedent will be added into the sub-rule-base:

$$P \ \& \ \prod_{i=1}^{m} M(i),$$

where m is the cardinality of M and $M(i)$ is the ith element of M. The output part of the rule will be an auxiliary output O_k representing the current matching conditions of the original rule. If the Q part of a disjunctive clause evaluates to false, it is still possible that overall disjunctive clause holds true as the Q' part of the clause might result in a true evaluation. For this reason, when an auxiliary output is generated, i.e., it is not possible to decide whether the conditions for the original rule hold true or not, this auxiliary output should be forwarded to the upper layers in the hierarchy so that the inputs available there can be used to further validate the conditions.

If we define the set M' as $M' = \{Q'_i \mid Q_i \notin M\}$ with the cardinality $(n - m)$, then a new rule with the original output O and the following antecedent is added into the central rule-base:

$$O_k \ \& \left(\prod_{i=1}^{n-m} M'(i) \right) \& \ R,$$

where $M'(i)$ is the ith element of M'.

If M and q are identical, the original rule is removed from the central rule-base. If R is missing in addition to that condition, the formula for adding a new rule into the newly generated rule-base does not differ from the previous case except when the rule will have the original output O rather than an auxiliary output as its output part.

The above process should be repeated for each member of the powerset $\mathbb{P}$ so that all possible combinations of condition matchings are enumerated. As a result of this process, a central rule is decomposed into multiple sub-rules and placed in two rule-bases: the newly created sub-rule-base and the modified central rule-base.

The above steps are for the decomposition of a single rule. In order to generate the complete sub-rule-base, the operations taken for just one rule should be repeated for every rule in the original rule-base. Furthermore, the described process only creates a sub-rule-base for one type of node and the complete distribution of information processing into the sensor network requires the creation of sub-rule-bases for every type of node residing in different hierarchies.

Example Let's consider $[(a_1 \ \& \ (b_1 \mid c_1) \ \& \ (d_1 \mid e_2 \mid f_2) \ \& \ (g_1 \mid h_2) \ \& \ i_2) \rightarrow O]$, where subscripts represent the level where the related input can be processed. For the sake of simplicity, the condition part of the rule is omitted. This rule is going to be distributed into two different rule-bases. The input set for the first level is IS = $\{a_1, b_1, c_1, d_1, g_1\}$. Using this, we come up with $P = (a_1 \ \& \ (b_1 \mid c_1))$, $q = \{d_1, g_1\}$, $q' = \{(e_2 \mid f_2), h_2\}$, and $R = i_2$.

The powerset $\mathbb{P}$ is equal to $\{\{\}, \{d_1\}, \{g_1\}, \{d_1, g_1\}\}$. If we follow the steps described in the algorithm, the following rules would be added into the new sub-rule-base:

$$(a_1 \ \& \ (b_1 \mid c_1)) \rightarrow X_1 \qquad (a_1 \ \& \ (b_1 \mid c_1) \ \& \ d_1) \rightarrow X_2$$
$$(a_1 \ \& \ (b_1 \mid c_1) \ \& \ g_1) \rightarrow X_3 \qquad (a_1 \ \& \ (b_1 \mid c_1) \ \& \ d_1 \ \& \ g_1) \rightarrow X_4$$

In accordance with these rules, the following rules are added into the original rule-base:

$$(X_1 \ \& \ (e_2 \mid f_2) \ \& \ h_2 \ \& \ i_2) \rightarrow O \qquad (X_2 \ \& \ h_2 \ \& \ i_2) \rightarrow O$$
$$(X_3 \ \& \ (e_2 \mid f_2) \ \& \ i_2) \rightarrow O \qquad (X_4 \ \& \ i_2) \rightarrow O$$

Finally, the original rule being decomposed is removed from the original rule-base.

3.3 Upper bound on the number of sub-rules

The main factors affecting the running time of RBDA and the number of sub-rules generated are the number of different node classifications, the number of rules in the original rule-base and the cardinality of the set q of each rule.

Let k be the number of different classifications, r be the number of rules in the original rule-base, n_i be the total number of sub-rules that are generated from the ith rule, n_i^k be the number of sub-rules for the kth classification that are generated from the ith rule and q_i^k be the cardinality of the set q of the ith rule for the kth classification.

For a typical sensor network, we expect the number of different classifications at different hierarchies to have a small value. For example, in a cluster-based homogeneous sensor network this number will most probably be 3. The cardinality of the set q changes with each rule and each classification. Similar to the number of classifications, we anticipate a small value for it on the average case.

The worst case in decomposing rules for a classification k occurs when there are P and R parts in each sub-rule generated for classification $k+1$. In such a case, the number of rules that are generated from one rule is as follows:

$$\begin{aligned}
n_i^1 &= 2^{q_i^1} \\
n_i^2 &= n_i^1 * 2^{q_i^2} \\
&= 2^{q_i^1} * 2^{q_i^2} \\
&\vdots \\
n_i^k &= n_i^k * 2^{q_i^k} \\
&= 2^{q_i^1} * 2^{q_i^2} * \cdots * 2^{q_i^k} \\
n_i^k &= \prod_{j=1}^{k} 2^{q_i^j} \qquad (1)
\end{aligned}$$

Using these, we can calculate the upper bound on the total number of sub-rules generated for the ith rule as follows:

$$n_i \leq \sum_{t=1}^{k} \left(\prod_{j=1}^{t} 2^{q_i^j} \right) = \sum_{t=1}^{k} 2^{\left(\sum_{j=1}^{t} q_i^j\right)} \qquad (2)$$

Therefore, the upper bound on total number of sub-rules are:

$$n = \sum_{i=1}^{r} n_i \qquad (3)$$

$$n \leq \sum_{i=1}^{r} \left(\sum_{t=1}^{k} 2^{\left(\sum_{j=1}^{t} q_i^j\right)} \right) \qquad (4)$$

Total number of sub-rules generated has exponential space complexity. Since rule-base decomposition is statically done outside of the sensor network, algorithm running time has little importance in terms of energy efficiency of the sensor network. However, number of rules influence both space and rule-processing time of sensor nodes. More rules being placed in a rule-base means that more time and computation required for processing them.

4 Storage-efficient rule decomposition

The main drawback of RBDA is that all possible n-ary combinations of the elements of the set q need to be enumerated. In such a case, the number of sub-rules generated grows exponentially depending on the number of elements in q. This situation results in two major problems. First, more rules means that more space is required for storage, yet storage is a luxury for a typical sensor node. Second, a larger rule-base leads to increased processing overhead as the information-processing engine needs to go over much more rules for the event detection and correlation operations.

In order to reduce the number of sub-rules generated, we extend RBDA with a new one that employs *rules with variables*, helping us to get rid of the need for statically listing all possible input combinations of a rule.

4.1 Rules with variables

We define the term predicate as any binary function that takes a variable number of arguments and returns a value of true or false. Furthermore, given a node's input set, we present a generalized format of the antecedent of the rules:

$$P \;\&\; (Q_1 \mid Q_1') \;\&\; (Q_2 \mid Q_2') \;\&\; \cdots \;\&\; (Q_n \mid Q_n') \;\&\; R,$$

where P, R, Q_i and Q_i' have the same meanings as described in the previous section.

Definition 4.1 Let q be the set $\{Q_i \mid 1 \leq i \leq n\}$ and q' be the set $\{Q_i' \mid 1 \leq i \leq n\}$. A variable is a symbol that represents some m-ary combination of the elements of the sets q or q', where $0 \leq m \leq n$.

Let's consider, for example, the following antecedents: P, $(P \;\&\; Q_1)$, $(P \;\&\; Q_2)$, $(P \;\&\; Q_1 \;\&\; Q_2)$. We can represent these in a single expression with the help of the variable: $(P \;\&\; \vartheta)$, where ϑ stands for either ϕ, Q_1, Q_2 or $(Q_1 \;\&\; Q_2)$. Using variables eliminates the requirement for exhaustively listing all possible input combinations.

Rules used in the new algorithm differ from the conventional rules in that additional array structures are associated with the variables. Array entries correspond to the elements of the sets q or q', or to the results of their evaluations. However, note that a variable may only be used for either the elements of set q or q', but not both. In other words, all entries of an array need to be related to the elements of the same set.

There are two roles that a variable can assume. The first role is to statically store the information about the disjunctive clauses to be matched, which are the elements of the sets q or q'. The second role is to keep and transfer dynamic knowledge about the results of the evaluation of the disjunctive clauses during rule processing. For this purpose, we define two types of variables:

1. *Predicate matching variable (PMV)* This is responsible for statically storing all possible disjunctive clauses that could be matched at run time. The rule engine uses PMV to determine what to search for in the available event stream at run time.
2. *Knowledge transfer variable (KTV)* This holds the results of the processing carried out by a node and is used by the nodes at the upper layers to determine what should be matched. As KTV stores the results of evaluations, its contents are meaningful only at run time. The antecedent of a rule cannot contain a KTV alone; it needs to be accompanied by a PMV. Actually, there is one-to-one relationship between the entries of the PMV and the KTV. The KTV contains the results of the evaluations of the elements of the set q

from previous layers, and the PMV stores the elements of the set q'. Variables in the consequent can only be KTV.

Sub-rules that are produced as a result of v-RBDA might contain zero, one or two variables in their antecedents.

- If there are no variables in a rule, the rule that is decomposed to produce that rule has an empty set q.
- If there is a single variable in a rule, it stores all elements of the set q. The variable is a PMV and this rule cannot be further decomposed. In the presence of such a rule, there must be an accompanying rule employed in the next (upper) layer. Rules generated for the nodes of the lowest layer cannot have more than one variable.
- If there are two variables in a rule, the first variable, which is a KTV, holds the knowledge about the processing carried out in the lower layer. The second variable is a PMV and it stores the elements of the set q' that is constructed using the input set of the lower layer. Besides keeping the relationship with the lower layer, the second variable also stores the elements of the set q, which is constructed for the current layer, such that processing in the current layer can be used to shape the processing carried out in the upper layers.

4.2 v-RBDA: decomposition using rules with variables

Similar to RBDA, given an input set for a node, v-RBDA decomposes a rule-base into two rule-bases: a new rule-base for the node (NRB), and the modified original rule-base (ORB).

Algorithm 2 gives a pseudocode implementation of the new rule-base decomposition process. It begins by putting the antecedent of the original rule into the conjunctive normal form. Each conjunct is either a single predicate or a disjunction of predicates. Conjuncts form P if all predicates can be evaluated by the node and conjuncts that cannot be evaluated form the R part of the antecedent. If the conjunct is in disjunctive normal form and only some part of it can be evaluated by the node, then the part that can be processed constitutes Q and the remaining part constitutes Q'. The antecedent of a rule can be written as $P\ \&\ (Q_1 \mid Q'_1)\ \&\ \cdots\ \&\ (Q_n \mid Q'_n)\ \&\ R$.

Let q be the set $\{Q_1,\ Q_2,\ \ldots,\ Q_n\}$ and q' be the set $\{Q'_1,\ Q'_2,\ \ldots,\ Q'_n\}$. If the rule to be decomposed only contains P, it is added into the NRB, and if it only contains R, it is added into the ORB unaltered. If the rule contains both P and R but not any $(Q \mid Q')$, a sub-rule with P as its antecedent and an auxiliary output as its consequent is added into the NRB. The associated rule to be placed in the RB has the conjunction of auxiliary output and R as its antecedent, together with the original rule's consequent as its consequent.

Provided that the set q is not empty, let n be the number of elements of the set q. With our new algorithm, instead of generating 2^n pairs of rules, a rule can be decomposed into at most three sub-rules regardless of the value of n: one or two rules for the NRB and one rule for the ORB. Two sub-rules, for the NRB, are generated when there is no R in the original rule and it is possible to reach a state where the original rule's actions can be taken. One of the rules contains the original rule's consequent, which covers the case that all elements of the set q are captured. And the other rule, which has an auxiliary output, is used to cover the cases in which not all the elements of the set q are captured.

Let ϑ represent the variable used in the sub-rule to be placed in the NRB and $\vartheta[\,]$ represent the array associated with it. If the set q is non-empty and the rule does not contain any variables, then the elements of q are assigned to the elements of the array associated with the variable:

$$\vartheta[i] = Q_i, \quad 1 \le i \le n$$

Algorithm 2 v-RBDA

```
1: INPUT: $IS$=input set, $RB$=original rule-base
2: OUTPUT: $RB$=original rule-base (modified), $NRB$=new sub-rule-base
3: for all $Rule$ in $RB$ do
4:     Set $P, R, q, q'$ ; $n \leftarrow$ cardinality of set $q$
5:     if $Rule$ does not contain variables then
6:         if (($R$ is NULL ) & ($P$ is not NULL | $n \geq 1$)) then
7:             Add $\left[(P \text{ \& } conjunction\_of\_all\_Q_i) \rightarrow O\right]$ into $NRB$
8:         else if $q$ is empty then
9:             if ($P$ is NULL) then
10:                Add $[R \rightarrow O]$ into $RB$
11:            else
12:                Add $[P \rightarrow O_p]$ into $NRB$
13:                Add $[(O_p \text{ \& } R) \rightarrow O]$ into $RB$
14:            end if
15:        else
16:            for $i = 1$ to $n$ do
17:                $\vartheta[i] = Q_i$ ; $\omega[i] = Q'_i$
18:            end for
19:            Add $[(P \text{ \& } \vartheta) \rightarrow \mu]$ into $NRB$
20:            Add $[(\mu \text{ \& } \omega \text{ \& } R) \rightarrow O]$ into $RB$
21:        end if
22:    else                              ▷ // $\left(P \text{ \& } \mu \text{ \& } \omega \text{ \& } (Q_1|Q'_1) \text{ \& } \cdots \text{ \& } (Q_n|Q'_n) \text{ \& } R\right)$
23:        $\vartheta \leftarrow remove\_non\_processable\_disjuncts(\omega)$
24:        $\omega \leftarrow remove\_processable\_disjuncts(\omega)$
25:        if ($R$ is NULL) then
26:            Add $\left[(P \text{ \& } \mu \text{ \& } \vartheta \text{ \& } conjunction\_of\_all\_Q_i) \rightarrow O\right]$ into $NRB$
27:        end if
28:        $k \leftarrow$ max index of $\vartheta$
29:        for $i = 1$ to $n$ do
30:            $\vartheta[k+i] = Q_i$ ; $\omega[k+i] = Q'_i$
31:        end for
32:        Add $[(P \text{ \& } \mu \text{ \& } \vartheta) \rightarrow \tau]$ into $NRB$
33:        Add $[(\tau \text{ \& } \omega \text{ \& } R) \rightarrow O]$ into $RB$
34:    end if
35:    Remove $Rule$ from $RB$
36: end for
```

If P is absent, only ϑ, otherwise P and ϑ constitute the antecedent of the sub-rule to be placed in the NRB. The rule has an auxiliary output, μ. It is a KTV and used as the first variable in the accompanying rule to be placed in the ORB. This rule has a second variable, ω, which is a PMV and holds the elements of the set q':

$$\omega[i] = Q'_i, \quad 1 \leq i \leq n$$

μ, ω, and, if it exists, R comprise the antecedent of the rule to be placed in the ORB. Its consequent is equal to the original rule's consequent. In addition to this pair of rules, if there is no R, then an additional rule, having the original rule's output as its consequent and the conjunction of the elements of the set q and, if it exists, P as its antecedent is added into the NRB.

If the rule to be decomposed contains variables, the generalized format for the antecedent must look like the following:

$$\left(P \text{ \& } \mu \text{ \& } \omega \text{ \& } (Q_1|Q'_1) \text{ \& } \cdots \text{ \& } (Q_n|Q'_n) \text{ \& } R\right),$$

where μ, a KTV, corresponds to the entries of the set q, and ω, a PMV, holds the entries of the set q'. Note, however, that these sets are not the ones used for the current decomposition;

they are created and used in the previous run of the decomposition algorithm. Let k be the size of the arrays associated with μ and ω. Let ϑ be assigned to ω with the non-processable predicates removed. Furthermore, let the processable predicates of ω be removed, leaving ω only with non-processable inputs. The array for ϑ will hold not only the elements coming from ω, but also the elements of the set q, which has the cardinality n:

$$\vartheta[k+i] = Q_i, \quad 1 \leq i \leq n$$

μ, ϑ and, if it exists, P constitute the antecedent of the sub-rule to be placed in the NRB. The rule has an auxiliary output, τ. It is a KTV and used as the first variable in the accompanying rule that is placed in the ORB. This rule uses ω as the second variable:

$$\omega[k+i] = Q_i', \quad 1 \leq i \leq n$$

τ, ω, and, if it exists, R comprise the antecedent of that rule. Its consequent is equal to the original rule's consequent. In addition to this pair of rules, if there is no R, then a rule with the original rule's output as its consequent and the conjunction of the elements of the set q, μ, ϑ and, if it exists, P as its antecedent is added into the NRB.

The last step is to remove the original rule from the ORB. Nevertheless, the steps described above decomposes only a single rule into sub-rules. These operations should be repeated for every rule in the ORB in order to generate the complete sub-rule-base for the type of node it is. The input set of this node is used in the algorithm. Furthermore, the described process only creates a sub-rule-base for one type of node; the complete distribution of information processing into the WSN requires the creation of sub-rule-bases for each type of node residing at different layers.

Example Let's consider the following rule for a three-level hierarchy:

$$a_1 \;\&\; (b_1 \mid c_2 \mid d_3) \;\&\; (e_1 \mid f_3)) \rightarrow O,$$

where subscripts represent the level where the related input can be processed. In this example, we are going to distribute this rule into three different rule-bases. In the first iteration of the algorithm, $P = a_1$, $q = \{b_1, e_1\}$, $q' = \{(c_2 \mid d_3), f_3\}$ and R is null. If we follow the steps described in the algorithm, the following rules would be added into the new sub-rule-base:

$$a_1 \;\&\; b_1 \;\&\; e_1 \rightarrow O$$
$$a_1 \;\&\; \vartheta[b_1, e_1] \rightarrow \mu[eval(b_1), eval(e_1)]$$

$eval(p)$ is the result of the evaluation of p. Elements of the KTVs are given for the sake of understandability. If all of a_1, b_1 and e_1 evaluate to true, then the rule engine reaches a conclusion and associated actions are taken. On the other hand, if at least one of b_1 or e_1 yields a false value or they cannot be evaluated because of the absence of input, then the available information is fused and the result is sent to the node in the next layer. After the first iteration, the original rule is removed from the ORB and the following rule is added into it:

$$\mu[eval(b_1), eval(e_1)] \;\&\; \omega[(c_2 \mid d_3), f_3] \rightarrow O$$

In the second iteration of the algorithm, sub-rule-bases for the second and third levels are generated by decomposing the above rule. The non-processable predicates in ω are d_3 and f_3. Based on this information, the sub-rule that is generated for the second level is as follows:

$$\mu[eval(b_1), eval(e_1)] \;\&\; \vartheta[c_2, -] \rightarrow \tau[eval(b_1) \mid eval(c_2), eval(e_1)]$$

After the second iteration, the original rule-base, which is the rule-base of the third level, has the following rule:

$$\tau[eval(b_1) \mid eval(c_2), eval(e_1)] \;\&\; \omega[d_3, f_3] \rightarrow O$$

4.3 Rule processing

In addition to the selection of rules that could fire, the rule engine must perform an additional task: matching variables with the elements of the sets q and q', which are the sets used during the decomposition of the rule that this rule is decomposed from.

The minimum requirements for a rule to be eligible for firing is that non-variable elements in the antecedent need to be detected. If the consequent is an auxiliary output, then not all of the elements in the array associated with the variable need to satisfy. The auxiliary output is used as a variable in the rules of the next layer.

In a rule which contains two variables, the first variable stores information about events that have been detected by the lower layers thus far. The second variable matches the remaining events using the information in the first variable. The second variable may also be used to pass the result of the partial processing, which has been performed up until the current layer, onto higher layers.

Let's consider the disjunctive clause $(Q \mid Q')$. Q is stored in the first variable and Q' is stored in the second variable. The index of the array entry that holds Q is equal to index of the entry that holds Q'. If the ith entry of the first variable evaluates to true, then the rule engine does not need to check the satisfaction of the ith entry of the second variable. Furthermore, if the second variable is used to pass the information onto higher layers, the ith entry is treated as if it has been satisfied. In other words, the ith entry of the second variable of the rule in the next layer does not need to be checked.

Let's consider the sub-rules from the example in 4.2. If a_1, b_1 and e_1 occur, then the rule in layer one fires and action O is taken. Now let's assume that events a_1, c_2 and f_3 have occurred. In the first layer, the second rule will fire even though ϑ does not match b_1 and e_1. Therefore, $\mu[false,\ false]$ is sent to the second layer. In the second layer, ϑ matches c_2 and therefore $\tau[true,\ false]$ is sent to the third layer. In the third layer, ω matches f_3 and the result of the evaluation becomes $\omega[true,\ true]$. As a result, the rule fires and action O is taken.

4.4 Upper bound on the number of sub-rules

The use of variables in rules helps us to transfer the burden of space into computation. The main factor that affects the number of sub-rules generated by the decomposition algorithm is the number of layers. In any layer except the highest layer, there can be at most two sub-rules generated from one rule. In the highest layer, there can be at most one rule.

Let k be the number of different layers, and n be the number of rules in the original rule-base. In the worst case, the number of sub-rules generated from a single rule will be less than or equal to $(2k - 1)$. Therefore, the upper bound on the total number of rules that can be used by the application is $n(2k - 1)$ (Fig. 1).

5 Application scenario

In-network processing of data is especially useful for event-driven applications where the focus is on events and reactions to them. Although some applications might require all

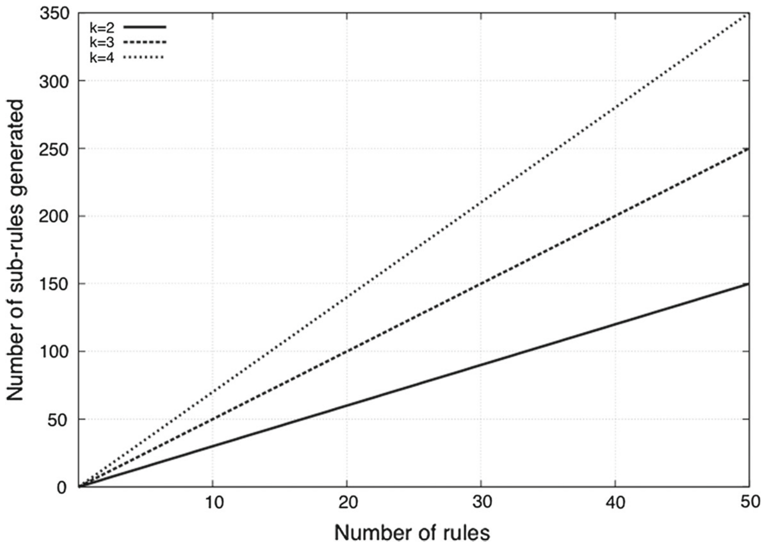


Fig. 1 Worstcase for the number of sub-rules generated

raw sensory data to be recorded, such as research-oriented applications where the aim is to extract knowledge about the inner workings of an unexplored real-world phenomenon, there are many application scenarios where the only interest is in high-level knowledge of whether a certain event happens or not. Early detection of forest fires and healthcare monitoring are typical examples for such applications.

We choose to discuss a healthcare monitoring application as a scenario that can benefit from our approach. In the following subsections, we give the details of the properties and requirements of such an application and then describe a system architecture that can be used for this purpose.

5.1 Properties and requirements

Healthcare monitoring has become an important application area for wireless sensor networks. The benefits of implementing such an application have been discussed in [4,14]. Nevertheless, there is still a lot to be done in the development of hardware and software before we see widespread implementations. For example, individuals would expect wearable sensors to be small, unobtrusive, harmless, reliable and long-lasting. In addition to these hardware issues, the requirements of timely and efficient data processing necessitate improvements in the underlying information processing and communication architectures.

In a healthcare monitoring application, a person's physiological signals, such as pulse rate, blood pressure, respiration rate and body temperature, together with physical activities and the state of individual and environmental conditions are assessed to detect and react to emergency cases. Take pulse rate as an example. A person's pulse rate while he is exercising might be twice its rate while he is resting. Besides, normal values of vital signs differ according to a person's age or sex. Because of this capacity for change, it is not enough to employ simple threshold-based filters to eliminate unnecessary network traffic. On the other hand, it is very important to relay only relevant information to the medical center as the person responsible

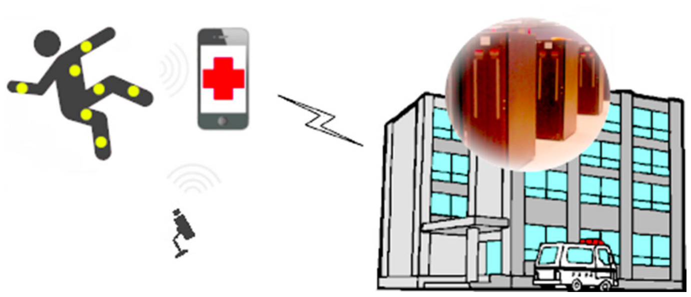


Fig. 2 Healthcare monitoring network

for monitoring and managing alerts might be overwhelmed by the number of alerts and miss important ones.

Another aspect of healthcare monitoring is that it requires continuous evaluation of sensor values; an emergency can happen at any time and in any place. If in-network processing is not used, all sensor readings would need to be transported to the data center.

Finally, time is very critical and immediate reaction is required in the case of an emergency. Therefore, the delay between the detection of an event and the reaction to it needs to be small. In this respect, in-network processing again performs better than central processing.

5.2 System architecture

The architecture of a healthcare monitoring application consists of a body area sensor network (BASN), a home network and a medical center network (Fig. 2).

Body area sensor network This consists of wearable medical sensors that sense the physiological signals of a person and sensor nodes that detect the posture and movement and/or other relevant physical activities or characteristics of the person. Medical sensors can detect pulse rate, blood pressure, respiration rate, body temperature, blood oxygen saturation and similar physiological signals. Additionally, physical motion sensors, such as the accelerometer and the gyroscope, are used to detect the current physical condition of the person [18]. An accelerometer is used to measure forward or upward acceleration so that it is possible to determine whether the person is running, walking, falling down or stationary. A gyroscope measures orientation, such as sitting, lying or standing. Wearable sensor nodes have limited processing capabilities, and they only check whether the sensed value is above or below a threshold value.

Gateway node This involves sensor nodes that communicate with a PDA or a special device that is used to collect and process the nodes' readings, act as a gateway between the BASN, home network and the medical center network, and react to emergencies. The gateway node also has capabilities to interact with the person being monitored. For example, if the sensor outputs are not enough to reach an accurate conclusion, an audiovisual alarm or an application might be activated that demands a confirmation response from the person. The inputs provided or not provided by the person can be recognized by the rule-processing engine as events, which then lead to other actions. Sensor nodes communicate with that device using 802.15.4 or a similar low power and a low data-rate protocol.

Home network Apart from wearable sensors, temperature or light sensors, IP cameras placed in the home might be used to evaluate environmental conditions, which can help clarify a person's state. They communicate with the gateway node.

Medical center network This contains a central server that stores and processes the information from individuals. Typical data that can be stored in central servers might be a person's medical history or results of a face-to-face examination. In addition to information-processing systems, there are also operators who are responsible for monitoring and managing the incoming information. The gateway node in a home network communicates with the medical center network using GSM, UMTS or similar mobile technologies.

6 Evaluation

We designed and performed comprehensive simulation experiments to evaluate our approach and algorithms. We first conducted a set of simulation experiments to see the effects of various parameters on the performance and behavior of our two proposed algorithms.Then we performed simulation experiments to evaluate our in-network processing approach. We measured the amount of reduction in total number of packets transmitted and energy consumed in the network when our approach is used. We compared the results of our approach with the results of centralized processing and directed diffusion.

6.1 Performance evaluation of the algorithms

We implemented both of our algorithms in a UNIX environment using C language. For the first experiment, we generated random rule-bases consisting of from 10 to 100 rules, for a two-layer decomposition. After repeating each experiment five times, each with different rule-bases, we plot the average of total number of all sub-rules for both layers.

For the second experiment, we used five different randomly generated rule-bases, each containing five rules, which may have at most five conjuncts and each of those conjuncts may contain at most four disjuncts. Throughout the experiment, we varied the input set of the node for each percentage value and took the average of the results for plotting.

For the third experiment, we used the same rules that we used in the second experiment, and we uniformly distributed inputs to the layers.

We used the rule-bases of the second and third experiments for evaluating the storage requirements of the algorithms. In order to calculate how much space is needed by each algorithm, we made the following assumptions. First, we assume that the predicates in the rules contain a pointer to the actual implementation of the predicate function. This knowledge is stored in two bytes (16 bits). Similar to predicates in the antecedent, we assume that the consequent part of the rules contains a pointer, two bytes, to the actual code that takes the necessary actions. A PMV requires as much space as the total space required by each predicate that could be matched by that variable. Finally, a KTV requires just two bytes, since a single bit is enough to represent the state of the element, and 16 bits is more than enough for a rule. In Table 1, RB-1 and RB-2 refer to the sub-rule-bases for layer 1 and layer 2 accordingly. The numbers seen in the table are the average of the number of bytes required for sub-rule-bases that are generated from the five original rule-bases. RBDA requires 90 bytes in total, whereas v-RBDA requires 27 bytes in total, which is 30 % of the space required for the former algorithm.

6.2 Performance evaluation of in-network processing

We also evaluated how much traffic and energy consumption reduction we can get in a WSN with the application of our approach. First, we want to discuss our energy consumption model

Table 1 Space needed by rule-bases

Algorithm	RB-1 (bytes)	RB-2 (bytes)
RBDA	47	43
v-RBDA	17	10

for a WSN. Then we present the network and application setup of our simulations. In the next section ("Results"), we present and discuss our results and compare our approach against centralized processing and directed diffusion, a well-known in-network processing scheme. In our simulations, we used Castalia simulator [7], which is a sensor platform independent WSN simulator based on OMNet++ platform.

6.2.1 Energy model

We consider sensor network with n sensor nodes, where the number of events detected by a sensor node has a poisson distribution with a mean event arrival rate of λ. Let h be the average number of hops for a node to reach the nearest cluster head, b be the number of bits in a network packet and c be the transmission rate in bits per second.

In a time interval $[0, t)$, λt events occur at a single node and $n\lambda t$ events occur throughout the entire network. The occurrence of a single event results in h transmission and $(h - 1)$ receive operations that are carried out by the sensor nodes. In order to calculate the energy consumed for these communication operations, we can use the formula

$$E = V \cdot I \cdot T, \tag{5}$$

where V is the potential difference, I is the electrical current and T is the time in seconds. In [34], the electrical current that flows through the Mica2's radio circuitry is listed under the presence of a 3 V power supply. The receive operation results in 7.0 mA current flow, whereas the transmission operation causes between 3.7 and 21.5 mA, depending on the power level used for the transmission. We can calculate the time required to transmit or receive a packet using the formula $T = b/c$.

The total energy consumed by the sensor nodes is determined by multiplying the energy required for the transmission and receive operations when an event occurs by the total number of events that occur in the sensor network:

$$E = \frac{3.n.\lambda.t.b}{c} \left(h \cdot I_t + (h - 1) \cdot I_r\right), \tag{6}$$

where I_t and I_r are the electrical currents that flow through the radio circuitry when transmitting and receiving packets, respectively.

In the above formula, the energy consumption that occurs while sensor nodes listen to the wireless channel is not taken into account. However, sensor nodes also need to power the radio electronics in a listening or idle state to detect the presence of radio signals. As mentioned in [37], the energy consumed during the listening state is big enough so as not to be ignored in the energy consumption analysis. Actually, we see from [34] that the same amount of current flows through the radio circuitry during the listen and receive operations.

If we assume that the same amount of energy is consumed while receiving and listening, then the total energy consumed by the sensor nodes will be:

$$E = 3.t. \left(\frac{n.\lambda.b.h.I_t}{c} + l.I_r\right), \tag{7}$$

Table 2 Power required for sensor node's operations

*O*peration	*P*ower (mW)
Receive	22.2
Idle	22.2
Transmit	80.1–15.9
Sense	0.02
Sleep	0.0006

where l is the ratio of listen time to the sum of sleep and listen times in a unit time interval. We choose to use this energy model in our simulations where we ignore the energy consumed during processing and sensing operations.

6.2.2 Network setup

In our setup, the sensor network consists of ordinary sensor nodes and highly capable cluster heads. Sensor nodes use cluster heads for the transmission of their data to the sink, and it is assumed that they can reach the cluster heads in just one hop. They choose the nearest cluster head as their next hop to sink. Similarly, cluster heads reach the sink in one hop. This means that any node in the sensor network can reach the sink in at most two hops.

The sensor field used in the simulations is $400\,\text{m} \times 400\,\text{m}$, and it is divided into 16 $100\,\text{m} \times 100\,\text{m}$ subregions. Each subregion contains one cluster head randomly deployed inside it. Simulations were run for 50, 100, 150 and 200 sensor nodes that are deployed uniformly in the field.

Table 2 shows the amount of power required for the operations carried out by Mica2 motes. As it can be seen from the table, a node in receive and idle states requires the same amount of power.

6.2.3 Application setup

Rules describing the application logic for this experiment are as follows:

$$\begin{aligned}
(x \geq 50) &\rightarrow A_1 \\
((last\ 5\ readings\ of\ y \geq 100)\ and\ (v \geq 500)) &\rightarrow A_2 \\
((x \geq 35)\ and\ (y \geq 50)) &\rightarrow A_3 \\
((x \leq 0)\ and\ (z \leq 0)\ and\ (w == 0)) &\rightarrow A_4 \\
((x \leq 0)\ and\ (z \leq 0)\ and\ (w == 1)) &\rightarrow A_5 \\
((y \geq 125)\ and\ ((w == 2)\ or\ (v < 400))) &\rightarrow A_6
\end{aligned}$$

These rules form the central rule-base. Ordinary sensor nodes sense x, y and z, whereas cluster heads sense v and w. The rules do not have any conditions.

We consider three different scenarios. In the first scenario, the sensory data are transmitted directly to the sink without any processing by the sensor nodes. Cluster heads act as gateways and they do not process data either. In order to reduce the total packet overhead and the number of packets, the x, y and z values are packed into a single packet. In the second scenario, both sensor nodes and cluster heads send data to the sink only if it is above or below some threshold value. This is similar to the case in directed diffusion [16], where data are sent if there is an interest in it. Sensory data meeting the threshold conditions are transmitted in

$$(not\_exercising \; \& \; (pr \leftarrow high \mid rr \leftarrow high \mid bp \leftarrow high)) \rightarrow \text{"}record\_abnormal\_event\text{"}$$
$$(exercising \; \& \; (pr \leftarrow low \mid rr \leftarrow low \mid bp \leftarrow low)) \rightarrow \text{"}record\_abnormal\_event\text{"}$$
$$(number\_of\_abnormal\_events \geq 3) \rightarrow \text{"}inform\_person\_and\_medical\_personnel\text{"}$$
$$(pr \leftarrow very\ low \mid rr \leftarrow very\ low) \rightarrow \text{"}inform\_medical\_personnel\text{"}$$
$$(pr \leftarrow high \; \& \; bp \leftarrow low) \rightarrow \text{"}ask\_person\_if\_he\_is\_OK\text{"}$$
$$(not\_OK\_signal \mid no\_response\_for\_active\_question) \rightarrow \text{"}inform\_medical\_personnel\text{"},$$

Fig. 3 Rules used in the healthcare monitoring application

a single network packet in this scenario too. Finally, in the last scenario, sensor nodes and cluster heads process data according to the rules in their rule-bases and only the results of the processing are sent to the sink.

6.3 Healthcare monitoring

In addition to the above simulations, we also conducted experiments for the healthcare monitoring application described in Sect. 5 to show that in-network processing is indispensable for event-driven applications. Here, the concern is not only minimizing network traffic and power consumption, but also generating timely and accurate reactions to important events.

6.3.1 Experiment setup

We used the rules in Fig. 3, where *pr* is pulse rate, *rr* is respiration rate, and *bp* is systolic blood pressure. The first four rules are self-explanatory. If the fifth rule fires, an application that gets the person's attention by audiovisual stimuli and asks him to provide his status is activated. That person is assumed to provide the necessary information if he is all right. Rule 6 covers the cases where there are no replies after rule has requested status information, or there is an explicit notification of a bad condition.

Wearable sensor nodes have a sampling interval of 30 s, and the simulation duration is 86,400 s, i.e., 1 day. Similar to the previous simulation, we consider both central and distributed processing. We consider three scenarios. In the first scenario, most of the time a person has normal values as his vital signs, but occasionally his pulse rate increases, while his blood pressure has a low value. In the second scenario, a person has high blood pressure values throughout the day. Finally, in the third scenario, a person having normal vital signs stops breathing for 2 min.

7 Results

In this section, we present the results of experiments described in the previous section.

7.1 Results for performance evaluation of the algorithms

In Fig. 4, we see how our algorithms behave for a varying number of rules in the initial rule-base. It is seen that the number of sub-rules generated by v-RBDA has a linear relationship with the original number of rules. The line is almost straight. Furthermore, the number of rules for a layer is close to the original number of rules. On the other hand, the number of sub-rules generated by RBDA depends entirely on the nature of the rules. Although random rule-bases

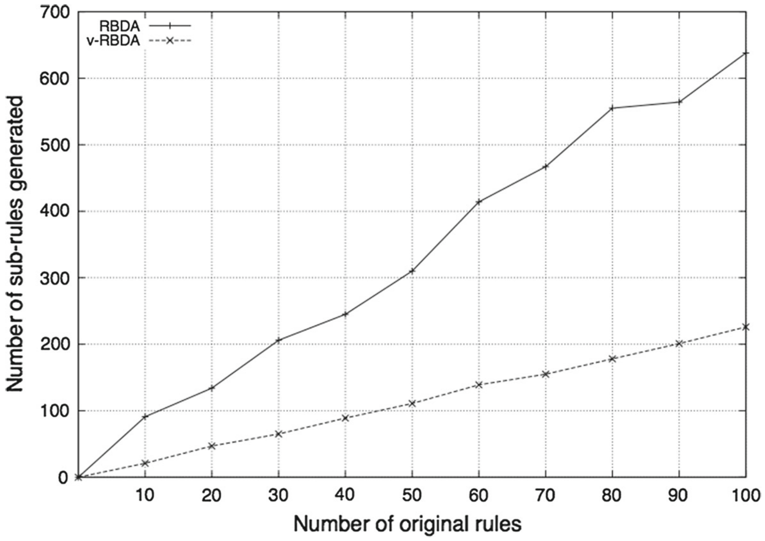


Fig. 4 Number of initial rules versus number of sub-rules generated

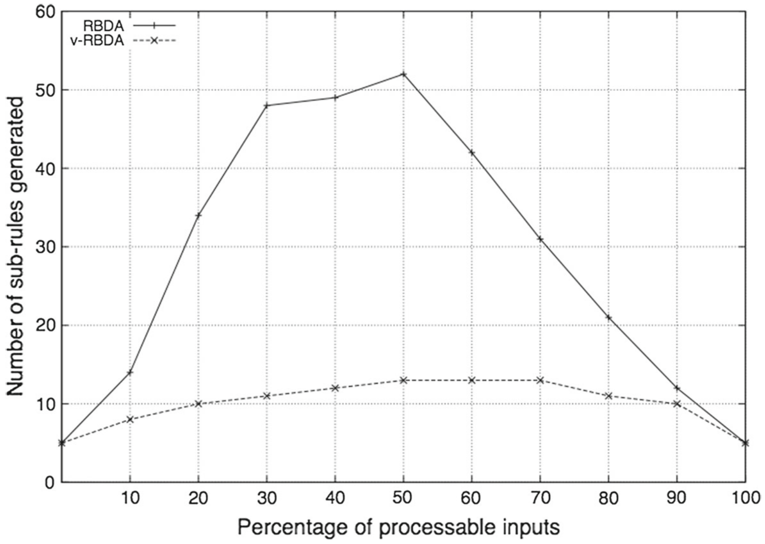


Fig. 5 Percentage of processable inputs versus number of sub-rules generated

have not generated the worst case, the difference between the results is still huge, especially if we consider the sensor nodes where energy conservation is of utmost importance.

In Fig. 5, we present the effect of percent of processable inputs of a node to total available inputs on the number of sub-rules generated. We can see from the results that there is a major difference in the number of sub-rules generated by RBDA and v-RBDA when the percentage is between 20 and 70. A percent of 0 means no processable inputs are present and 100 means all inputs can be processed. In these cases, the original number of rules is preserved.

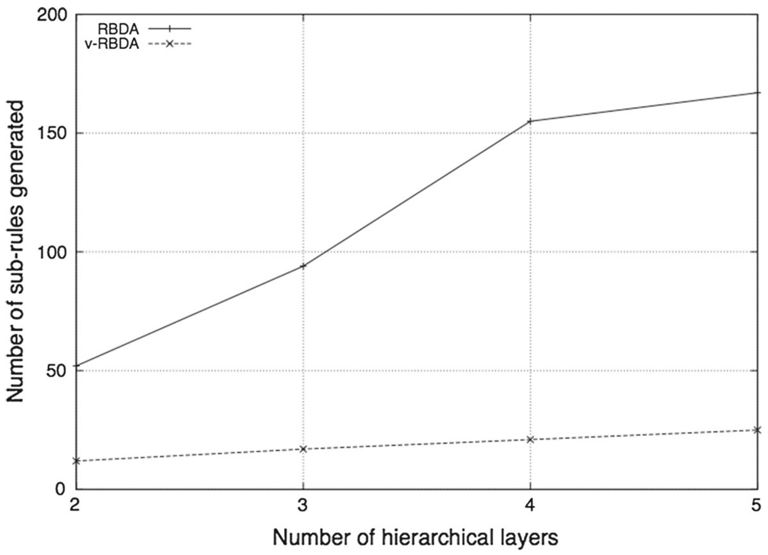


Fig. 6 Number of different layers versus number of sub-rules generated

In Fig. 6, we see the effect of the number of different layers, which is equal to the number of different classifications, on the number of sub-rules generated. The results lead us to a similar conclusion as the previous experiment, where, with everything being equal, v-RBDA produces fewer rules, and compared to RBDA, the number of layers has less influence and the number of sub-rules generated is more predictable.

7.2 Simulation results for performance evaluation of in-network processing

Simulations were run for 5000 s for each of the application scenarios under exactly the same conditions; that is, the same network topology, the same energy and the physical event models were used in all of the application scenarios. Figures 7 and 8 show the simulation results.

Figure 7 shows the total number of packets sent by the sensor nodes. The results in the figures do not include the packets transmitted by cluster heads. As it can be seen from the figure, the total amount of communication is drastically reduced by employing our approach, used in the third scenario. In addition to the decrease in the number of packet transmissions, the packet size for the third scenario is smaller compared to the packet sizes for the other two scenarios. In the first two scenarios, raw sensory data, together with its temporal and spatial properties, are transported inside the network; in the third scenario, only the current state of the processing is propagated to the appropriate nodes. An eight-bit data can represent 2^8 different states. Additionally, due to spatial and temporal locality properties of information processing in WSNs, spatial and temporal information is either not transported or is transported in a short path in the network. Therefore, the reduction in the total number of bytes transmitted is much lower than the reduction in the total number of packets. In the simulations, we assumed that no collisions occur in the network. Had collisions been taken into account, the results for the first and second applications would have been worse due to the fact that more network traffic would lead to higher probabilities of packet collisions and retransmissions.

Figure 8 shows the average amount of energy consumed by each sensor node. As seen from the figure, the total spent energy is decreased by nearly a factor of three when the data

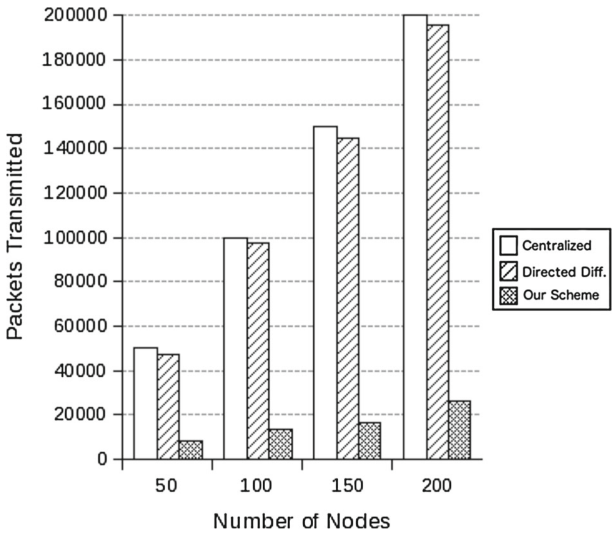


Fig. 7 Total number of packets transmitted by sensor nodes

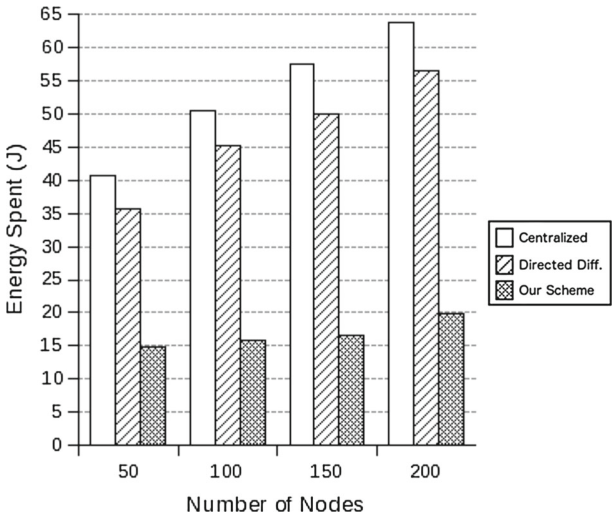


Fig. 8 Average energy consumption by sensor nodes

are processed inside the network using our approach. However, the number of transmitted packets is reduced by nearly a factor of seven. The nodes' energy consumption is not a corresponding seven times lower because they use the same amount of energy in the idle state as in the receive state.

In all of the scenarios, only one-hop clusters exist and cluster heads reach the sink in just one hop. In the case of multi-hop clusters, we anticipate better results with in-network processing compared to other approaches, because in such a case, nodes need to route other nodes' packets as well. The more network packets put into the sensor network, the more a node needs to transmit the packets of others. In addition to the obvious implications of power consumption, more packets cause more collisions and more retransmissions.

7.3 Results for healthcare monitoring application

The simulation results can be seen in Table 3. Link 1 refers to communication between the sensor nodes and the gateway node, and Link 2 is for the communication between the gateway and the central network. The results confirm the expectations that in-network processing makes a big difference. Instead of transmitting thousands of event notifications to a central network, only necessary events are transmitted.

Another parameter to look for is the delay in reacting to emergency cases. If the fifth rule is evaluated at a centralized system and it fires, the reaction requires getting back to home network and activating an application that interacts with the person. Table 4 shows how many hops are required for such a rule to reach a conclusion that the situation requires emergency personnel's response. It is wise to support decisions reached by bare sensor readings with a confirmation response from people in order to reduce false alarms. The case in the fourth row in the table would be frequently encountered in such a scenario. That row indicates that a decision can be made at the gateway node, but with central processing data need to be transported to central network. Furthermore, action, i.e., soliciting a confirmation request from the user, can only be taken at the gateway node. For central processing, transferring data from the sensor nodes to the gateway node, from the gateway node to the central system, from the central system back to the gateway (user confirmation request), and finally from the gateway to the central system (user's response) sum up to four hops. On the other hand, if distributed processing is allowed, sensor readings are transferred only to the gateway, and the rest of the processing is carried out at this node.

8 Discussion

In this paper, we focus on reducing the number of network transmissions by processing data inside the network for event-driven sensor network applications. Experimental results show

Table 3 Number of packet transmissions for healthcare monitoring

Scenario	Centralized		Distributed	
	Link 1	Link 2	Link 1	Link 2
Case 1	11,520	2896	2862	3
Case 2	11,520	3926	3025	173
Case 3	11,520	2880	2508	4

Table 4 Number of hops required for a decision

Location		Number of hops	
Firing	Action	Centralized	Distributed
Center	Center	2	2
Center	Gateway	4	3
Gateway	Center	2	2
Gateway	Gateway	4	1

that the number of transmitted packets and in turn the amount of energy consumed by sensor nodes are reduced by employing our strategy compared to central processing and directed diffusion. In fact this is the expected behavior, because if a sensor reading results in the antecedent of a rule being false in the first place, there is no need to carry this and other related data in the context of the rule.

Our strategy is going to perform better in the presence of sensor nodes that can sense multiple stimuli. In a sensor network where all sensor nodes sense a single stimuli, the reduction in network packets will be achieved by only not forwarding data that will lead to a failed rule antecedent. In such an environment, a reading might still result in an unnecessary transmission, because in the hierarchy of data processing the current value is not enough to determine the outcome of a rule, whereas other readings up in the hierarchy lead to the failed rule. If sensor nodes sense more than one phenomenon, in addition to previous achievement, multiple data readings will be fused into a single value. Additionally, the probability of unnecessary packet transmissions will decrease.

We present two schemes: RBDA and v-RBDA. This is because the first scheme is the more intuitive and easy-to-understand way to deal with the problem. However, regarding the resource limitations of sensor nodes, the number of rules generated as a result of rule-base decomposition becomes a major issue that need to be tackled. Therefore, we introduced a new rule representation and adapted the first method so that it can handle this new representation. However, if sensor network application does not have a complex rule-base, the first scheme can still be the choice due to its simplicity. For complex and large rule-bases, the second scheme has to be employed.

Since rule-base decomposition is carried out outside of sensor networks and on rare occasions like initial deployment or when there is a need for changing the logic of an application, its execution time is less important compared to the space requirements of sub-rule-bases, because the execution time is nothing to do with sensor nodes and the size of rule-bases directly affect them. Although technological advancements may make storage capacity being a smaller concern in the future, it is currently a limitation for small sensor nodes.

9 Conclusions

Our motivation in this paper is to process data inside sensor network as much as possible since communication operations are responsible for most of the energy consumption in sensor nodes. Furthermore, in-network processing might make a difference in early response for time-critical applications like healthcare monitoring application. We present two schemes that distribute information processing into different nodes in a wireless sensor network. For each distribution scheme, we present how a central rule-base expressing the application logic is decomposed into multiple sub-rule-bases, which are going to be employed in appropriate places inside the network. Our first method, using ordinary ECA rules, shows a simple, easy-to-grasp and rational way of distributing information processing. However, in the case of complex rule-bases, it suffers from an exponential increase in the number of sub-rules generated. Therefore, we introduce a new way of representing rules and modify sub-rule-base generation algorithm accordingly. We make the upper-bound analysis of both algorithms regarding the number of sub-rules generated and perform experiments, results of which confirm that the second method removes the deficiencies of the first method.

We describe an application scenario in order to show the application types that are suitable for our in-network processing approach. In the performance evaluation section, we provide the results of simulations that show the number of packets to be transmitted and the energy to be spent while processing sensor data. Based on our simulations we can see that whether an application employs our in-network processing approach, the average energy consumption of a sensor node might be reduced to nearly one-third of the energy consumption compared to cases when a central processing or simple threshold filtering approach is used.

Acknowledgements This work is supported in part by a research Grant from TUBITAK with Grant No. 114R082.

Appendix 1: Proof of RBDA

Before we proceed with the proof of the algorithm, let's give the common facts, terms and notations used in this subsection.

By a single operation of the algorithm, a rule is decomposed into sub-rules, which are then to be processed by two different types of nodes. The node whose input set is used to determine P, Q, Q' and R is referred to as the first node and the other as the second node. The antecedent of the original rule might contain P, $(Q_i \mid Q'_i)$, R or any combination of them. There are three cases that this could possibly fit:

– There is no $(Q \mid Q')$ in the rule.

  - **Case 1** There is either P or R. If there is P, $r : P$ is placed in the first node's rule-base, and if R, $r : R$ is placed the second node's rule-base.
  - **Case 2** There are both P and R. A single pair of rules is generated: $r_1 : P$, which is placed in the first node's rule-base, and $r_2 : r_1 \ \& \ R$, which is placed in the second node's rule-base.

– **Case 3** There are $(Q \mid Q')$ and possibly P and/or R. If there are n conjuncts in the form of $(Q \mid Q')$, there can be $n + 1$ cases that we need to consider: 0 to n Qs being detected by the first node. For each of $n + 1$ cases, there are $C(n, i)$ different combinations of Qs where $0 \leq i \leq n$. Therefore, there can be a total of $\sum_{i=0}^{n} C(n, i) = 2^n$ different cases that can be encountered during run-time processing. The rule-decomposition algorithm exhaustively generates a distinct pair of rules for every possible case. Let q be the set $\{Q_1, Q_2, \ldots, Q_n\}$, q' be the set $\{Q'_1, Q'_2, \ldots, Q'_n\}$, s be any subset of q and s' be the subset of q' such that $(Q'_i \in s') \Leftrightarrow (Q_i \notin s)$. The antecedents of a pair of rules r_1 and r_2 generated by the rule-decomposition algorithm are as follows (coe(x) means conjunction of elements of x):

  - $r_1 : \ (P \ \& \ {<}coe(s){>})$, which is placed in the first node's rule-base and
  - $r_2 : \ \big({<}output\ of\ r_1{>} \ \& \ {<}coe(s'){>} \ \& \ R\big)$, which is placed in the second node's rule-base.

  If, however, there is no R and the case considered is for n Qs, only a single rule is added into the first node's rule-base: $r : \ (P \ \& \ {<}conjunction\ of\ elements\ of\ q{>})$. For any pair of r_1 and r_2, the union of the conjuncts of r_1 and r_2 covers all the conjuncts of the original rule, since s together with s' cover all conjuncts that are in the form $(Q \mid Q')$ of the original rule.

Lemma 1 *An event pattern captured by a rule is also captured by the sub-rules, which are decomposed from that rule.*

Proof We need to prove that if original rule satisfies, r or r_2 also satisfies. Consider the three cases one by one:

1. *P or R* As the original rule and the rule placed in the sub-rule-base are the same, the same events are detected.
2. *P and R* Satisfaction of the original rule means that P and R evaluate to true. If P satisfies, r_1 also satisfies. Similarly, the satisfaction of r_1 and R results in the satisfaction of r_2. Therefore, if the original rule satisfies, sub-rules r_1 and r_2 also satisfy.
3. *Existence of* $(Q \mid Q')$ Satisfaction of the original rule implies that P, R and each $(Q \mid Q')$ evaluate to true. Since an exhaustive list of possible cases enumerated by sub-rules, at least one rule pair out of 2^n pairs will catch the event pattern detected by the original rule. □

Lemma 2 *An event pattern that cannot be captured by a rule cannot be captured by the sub-rules generated as a result of the decomposition of that rule.*

Proof We need to prove that if the original rule does not satisfy, neither does r or r_2. Consider the three cases one by one:

1. *P or R* As the original rule and the rule placed in the sub-rule-base are the same, the same event pattern should yield the same result.
2. *P and R* If the original rule does not satisfy, either P or R should yield a false evaluation. If P were false, r_1 and, as a result, r_2 would not satisfy. If R were false, r_2 would not satisfy. Therefore, if the original rule does not satisfy, neither does sub-rule r_2.
3. *Existence of* $(Q \mid Q')$ If the original rule does not satisfy, there must be at least one conjunct that evaluates to false. Since the conjuncts of the original rule are covered by all pairs of sub-rules, each pair of sub-rules should have at least one conjunct that does not satisfy, which means that if the original rule does not satisfy, sub-rules do not satisfy either. The same reasoning applies to the case in which there is only r, which covers all conjuncts of the original rule.

Theorem 1 *For a two-layer hierarchy, sub-rules that are decomposed from a rule capture exactly the same events as the original rule.*

Proof In order for this to be true, the following conditions should hold:

– an event pattern should be captured by sub-rules if and only if it could be captured by the original rule,
– an event pattern should not be captured by sub-rules if and only if it could not be captured by the original rule.

Let $A(e)$ be a proposition expressing an event pattern e detected by the original rule, $B(e)$ be a proposition expressing e detected by the sub-rules of that rule, $A'(e)$ be the proposition expressing e not detected by the original rule, and $B'(e)$ be the proposition expressing e not detected by the sub-rules. We need to prove that $(A(e) \Leftrightarrow B(e))$ and $(A'(e) \Leftrightarrow B'(e))$. In order to prove $(A(e) \Leftrightarrow B(e))$, we need to show $(A(e) \Rightarrow B(e))$ and $(B(e) \Rightarrow A(e))$. Similarly, in order to prove $(A'(e) \Leftrightarrow B'(e))$, we need to show $(A'(e) \Rightarrow B'(e))$ and $(B'(e) \Rightarrow A'(e))$. However, we know from propositional logic that $(A(e) \Rightarrow B(e)) \vdash (B'(e) \Rightarrow A'(e))$ and $(A'(e) \Rightarrow B'(e)) \vdash (B(e) \Rightarrow A(e))$. Therefore, it is enough to show that $(A(e) \Rightarrow B(e))$ and $(A'(e) \Rightarrow B'(e))$. The proofs for these are given in Lemmas 1 and 2. □

Theorem 2 *For a k-layer hierarchy, sub-rules that are decomposed from a rule capture exactly the same events as the original rule.*

Proof (Proof by Induction) A single operation of the decomposition algorithm generates two sub-rule-bases: a new rule-base for the node whose input set is used by the algorithm, and a modified original rule-base (MORB).

- *Initial Step* For two-layer hierarchy, a single running of the algorithm generates two sub-rule-bases. The sub-rules of a decomposed rule capture exactly the same events as the original rule, as proved by Theorem 1.
- *Inductive Step* For a k-layer hierarchy, the decomposition algorithm should be run $(k-1)$ times. We have to prove that for an arbitrary $k \geq 3$, if the theorem holds for $(k-1)$ sub-rule-bases, it also holds for k sub-rule-bases. Let's assume that the algorithm is run $(k-2)$ times, which generates $(k-1)$ rule-bases, and the events detected by the original rule are exactly the same as the events detected by sub-rules distributed into $(k-1)$ rule-bases. In order to generate the final rule-bases for $(k-1)$th and kth layers, the decomposition algorithm should be run once more, taking the following inputs: the input set of the $(k-1)$th layer and the MORB, generated from the $(k-2)$th iteration of the algorithm. Theorem 1 proves that sub-rules that are decomposed from a rule in MORB and placed in the newly generated rule-bases capture exactly the same event patterns as the event patterns detected by that decomposed rule. Based on this, we can conclude that sub-rules of an initial rule placed in k rule-bases capture exactly the same events as that rule does, since the event patterns captured by the rules of first $(k-2)$ together with the patterns captured by MORB constitute the event pattern of the initial rule. □

Appendix 2: Proof of v-RBDA

By a single operation of the algorithm, a rule is decomposed into sub-rules to be processed by two different types of nodes. Let's refer to the rule-base of the node where the input set is used to determine P, Q, Q' and R as the first rule-base and the other one as the second rule-base.

The antecedent of the original rule might contain P, a KTV/PMV pair, $(Q_i \mid Q'_i)$, R or any combination of these. It might contain no or two variables in its antecedent: a KTV (corresponding to the elements of q) and a PMV (corresponding to the elements of q'). They are related such that if the consequent of a rule does not contain a variable (i.e., it is a decision rule) and the non-variable conjuncts have been satisfied, the following condition must hold in order for the rule to fire: $\forall i$, $(\text{KTV}[i] \mid \text{PMV}[i]) \rightarrow \text{true}$.

There are four cases that the original rule could possibly be in:

- There are neither variables nor $(Q \mid Q')$ in the rule.
  - **Case 1** There is either P or R. If there is P, $r : P$ is placed in the first rule-base and if R, $r : R$ is placed the second rule-base.
  - **Case 2** There are both P and R. A single pair of rules is generated: $r_1 : P$, which is placed in the first rule-base, and $r_2 : r_1 \ \& \ R$, which is placed in the second rule-base.
- **Case 3** There are $(Q \mid Q')$ and possibly P and/or R in the rule, but no variables. Let ϑ and μ correspond to the elements of q, and ω corresponds to the elements of q'. A pair of rules, r_1, for the first rule-base, and r_2, for the second rule-base, are as follows: $r_1 : ([P \ \&] \ \vartheta) \rightarrow \mu$, $r_2 : (\mu \ \& \ \omega \ [\& \ R]) \rightarrow O$. If there is no R, an additional rule is

added into the first rule-base: r : ([P &]<$conj\_of\_elements\_of\_q$>) $\rightarrow O$. Actually, r_1 and r_2 could detect the same event patterns as r, but its sole purpose is to enable the first node to react to events.
- **Case 4** There is a KTV/PMV pair and possibly $(Q \mid Q')$ and/or P and/or R in the rule. Let KTV be μ and PMV be θ for the original rule, ϑ and τ correspond to the elements of q and processable entries of θ, and ω correspond to the elements of q' and the non-processable elements of θ. The antecedents of a pair of rules r_1 and r_2 generated by the rule-decomposition algorithm are as follows: r_1 : ([P &] μ & ϑ) $\rightarrow \tau$, r_2 : (τ & ω [& R]) $\rightarrow O$. If there is no R, an additional rule is added into the first rule-base: r : ([P &] μ & ϑ [&<$conj\_of\_elements\_of\_q$>]) $\rightarrow O$. It is used for enabling the first node to react to events.

An event pattern detected by the original rule is also detected by sub-rules if r_2 or r fires. Because an event pattern that is detected by r is also detected by r_1, we will omit discussing it in the rest of the proof. Furthermore, as the proofs for cases 1 and 2 are similar to the ones given in Sect. 3, here we discuss only cases 3 and 4.

Lemma 3 *An event pattern captured by the original rule is also captured by sub-rules, which are decomposed from that rule.*

Proof Let $E(Pr)$ represent the result of the evaluation of Pr. The firing of the original rule means that all conjuncts satisfy; in other words, $E(P)$, $E(R)$, $\forall i \;\; E(Q_i \mid Q'_i)$ and $\forall i \;\; E(\mu[i] \mid \theta[i])$ (only for case 4) are true.

Case 3 As $E(P)$ is true, r_1 fires with $\mu[i]$ set to true if $E(Q_i)$ is true, or else it is set to false. Since all $(Q_i \mid Q'_i)$ evaluate to true, if $\mu[i]$ (=$E(Q_i)$) is not true, $\omega[i]$ (=$E(Q'_i)$) needs to satisfy. As a result, the condition that "$\forall i,$ (KTV[i] | PMV[i]) $\rightarrow$ true" holds and because $E(R)$ is true, r_2 satisfies.

Case 4 Let there be x conjuncts in the form of $(Q \mid Q')$ and the number of entries of μ be y. The first y elements of ϑ contain the disjunctions of the processable predicates of θ and the remaining x elements correspond to Q_i. Similarly, the first y elements of ω contain the disjunctions of the non-processable predicates of θ and the remaining x elements correspond to Q'_i. τ contain the results of the evaluations of the elements of ϑ. As $E(P)$ is true, r_1 fires, with $\tau[i]$ is set to true if $\mu[i]$ is true or $\vartheta[i]$ satisfies, or else it is set to false. As all $(Q_i \mid Q'_i)$ evaluate to true, the disjunction of $\tau[i]$ and $\omega[i]$, where $(y+1) \leq i \leq (y+x)$, needs to produce true as well. Since $\theta[i] = \vartheta[i] \mid \omega[i]$ for $1 \leq i \leq x$, what is captured by θ in the original rule will also be captured by the τ and ω variable pair. As a result, the condition that "$\forall i,$ (KTV[i] | PMV[i]) $\rightarrow$ *true*" holds and because $E(R)$ is true, r_2 satisfies. □

Lemma 4 *An event pattern not captured by the original rule is not captured by the sub-rules, which are decomposed from that rule, either.*

Proof If the original rule does not satisfy, at least one of P, R, $(Q_i \mid Q'_i)$ or $\theta[i]$ (only for case 4) should fail to satisfy. If $E(P)$ is false, r_1 and in turn, r_2 do not fire for cases 3 and 4. Similarly, if $E(R)$ is false, r_2 fails to satisfy. If $E(P)$ and $E(R)$ are true:

- *Case 3* As the original rule fails to fire, the following condition should hold: $\exists i, \;\; E(Q_i \mid Q'_i) \rightarrow$ false. In such a case, although r_1 fires, r_2 fails to fire since the condition that "$\forall i$ (KTV[i] | PMV[i]) is true" does not satisfy.
- *Case 4* If there exists at least one i for which $E(Q_i \mid Q'_i)$ is false, r_2 does not fire for the reason explained for case 3. Otherwise, the following condition must be true: $\exists i, (\mu[i] \mid \theta[i]) \rightarrow false$. Since $\theta[i] = \vartheta[i] \mid \omega[i]$, what is captured by θ in the original

rule will also be captured by the τ and ω variable pair in r_2. As a result, the condition that "$\forall i$, (KTV[i] | PMV[i]) $\rightarrow$ true" would fail and therefore r_2 fails to satisfy.

As we can see, it is not possible to fire r_2 if the original rule fails to fire. □

Theorem 3 *For a two-layer hierarchy, sub-rules that are decomposed from a rule capture exactly the same events as the original rule.*

Proof In order for this to be true, the following conditions should hold:

- an event pattern should be captured by sub-rules if and only if it could be captured by the original rule,
- an event pattern should not be captured by sub-rules if and only if it could not be captured by the original rule.

The proofs given for Lemmas 3 and 4 demonstrate that these conditions hold true.

Theorem 4 *For a k-layer hierarchy, sub-rules that are decomposed from a rule capture exactly the same events as the original rule.*

Proof The proof is similar to the proof of Theorem 2. □

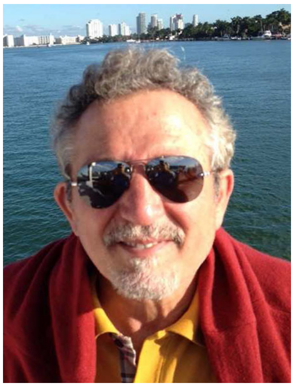
References

1. Ahn S, Kim D (2006) Proactive context-aware sensor networks. Lect Notes Comput Sci 3868/2006:38–53
2. Akyildiz IF, Weilian S, Sankarasubramaniam Y, Cayirci E (2002) A survey on sensor networks. IEEE Commun Mag 40(8):102–114
3. Apiletti D, Baralis E, Cerquitelli T (2011) Energy-saving models for wireless sensor networks. Knowl Inf Syst 28(3):615–644
4. Bonato P (2010) Advances in wearable technology and its medical applications. In: Proceedings of 32nd annual international conference of the IEEE engineering in medicine and biology society (EMBC), Buenos Aires, Argentina, September 2010
5. Braginsky D, Estrin D (2002) Rumor routing algorithm for sensor networks. In: Proceedings of the 1st ACM international workshop on wireless sensor networks and applications, Atlanta Georgia, USA, September 2002, pp 22–31
6. Carrano RC, Passos D, Magalhaes LCS, Albuquerque CVN (2014) Survey and taxonomy of duty cycling mechanisms in wireless sensor networks. IEEE Commun Surv Tutor 16(1):181–194
7. Castalia: a simulator for WSNs. http://castalia.forge.nicta.com.au/. Last Accessed on Feb 2016
8. Chong SK, Gaber MM, Krishnaswamy S, Loke SW (2011) Energy conservation in wireless sensor networks: a rule-based approach. Knowl Inf Syst 28(3):579–614
9. Eckert M (2008) Complex event processing with XChange^{EQ}: language design, formal semantics, and incremental evaluation for querying events, Ph.D. Dissertation, October 2008
10. Fei X, Magill E (2012) REED: flexible rule based programming of wireless sensor networks at runtime. Comput Netw 56(14):3287–3299
11. Fragouli C, Widmer J, Boudec J-Y L (2006) A network coding approach to energy efficient broadcasting: from theory to practice. In: Proceedings of IEEE INFOCOM 2006, Barcelona, Spain, April 2006, pp 1–11
12. Gatziu S, Dittrich KR (1994) Detecting composite events in active databases using petri nets. In: Proceedings of the 4th international workshop on research issues in data engineering: active database systems, pp 2–9, February 1994
13. Hande A, Polk T, Walker W, Bhatia D (2006) Self-powered wireless sensor networks for remote patient monitoring in hospitals. Sensors 6 9:1102–1117
14. Hanson MA, Powell HC, Barth AT, Ringgenberg K, Calhoun BH, Aylor JH, Lach J (2009) Body area sensor networks: challenges and opportunities. Computer 42(1):58–65
15. Hung C-C, Peng W-C (2011) Optimizing in-network aggregate queries in wireless sensor networks for energy saving. Data Knowl Eng 70(7):617–641
16. Intanagonwiwat C, Govindan R, Estrin D (2000) Directed diffusion: a scalable and robust communication paradigm. In: Proceedings of the 6th annual international conference on mobile computing and networking, Boston MA, USA, August 2000, pp 56–67

17. Jiao B, Son S, Stankovic J (2005) GEM: generic event service middleware for wireless sensor networks. In: Proceedings of the 2nd international workshop on networked sensing systems (INSS), San Diego, CA, USA, June 2005
18. Khan AM, Lee Y-K, Lee SY, Kim T-S (2010) A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. IEEE Trans Inf Technol Biomed 14(5):1166–1172
19. Kotidis Y (2005) Snapshot queries: towards data-centric sensor networks. In: Proceedings of the 21st international conference on data engineering, Tokyo, Japan, April 2005, pp 131–142
20. Lai Y, Zeng W, Lin Z, Li G (2010) LAMF: framework for complex event processing in wireless sensor network. In: Proceedings of the 2nd international conference on information science and engineering, Hangzhou, China, December 2010, pp 2155–2158
21. Lara R, Benitez D, Caamano A, Zennar M, Rojo-Alvarez JL (2015) On real-time performance evaluation of volcano-monitoring systems with wireless sensor networks. IEEE Sens J 15(6):3514–3523
22. Li M, Liu Y (2009) Underground coal mine monitoring with wireless sensor networks. ACM Trans Sens Netw 5(2):1–29
23. Liu X, Cao J, Tang S, Guo P (2015) Fault tolerant complex event detection in WSNs: a case study in structural health monitoring. IEEE Trans Mob Comput 14(12):2502–2515
24. Lloret J, Garcia M, Bri D, Sendra S (2009) A wireless sensor network deployment for rural and forest fire detection and verification. Sens 9 9(11):8722–8747
25. Lu G, Sadagopan N, Krishnamachari B, Goel A (2005) Delay efficient sleep scheduling in wireless sensor networks. In: Proceedings of IEEE INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies, vol 4, Miami, FL, USA, March 2005, pp 2470–2481
26. Luo H, Ye F, Cheng J, Lu S, Zhang L (2005) TTDD: two-tier data dissemination in large-scale wireless sensor networks. Wirel Netw J 11(1–2):161–175
27. Ma J, Lou W, Wu Y, Li X-Y, Chen G (2009) Energy efficient TDMA sleep scheduling in wireless sensor networks. In: Proceedings of IEEE INFOCOM 2009, Rio de Jenairo, Brazil, April 2009, pp 630–638
28. Madden SR, Franklin MJ, Hellerstein JM, Hong W (2005) TinyDB: an acquisitional query processing system for sensor networks. ACM Trans Database Syst 30(1):122–173
29. Manjeshwar A, Agrawal DP (2000) TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In: Proceedings of 15th international parallel and distributed processing symposium, San Francisco, CA, USA, April 2000, pp 2009–2015
30. Marcelloni F, Vecchio M (2008) A simple algorithm for data compression in wireless sensor networks. IEEE Commun Lett 12(6):411–413
31. Perianu MM, Havinga P (2007) D-FLER: distributed fuzzy logic engine for rule-based wireless sensor networks. Ubiquitous Comput Syst 4836(2007):86–101
32. Pietzuch PR, Shand B, Bacon J (2004) Composite event detection as a generic middleware extension. In: IEEE network magazine, special issue on middleware technologies for future communication networks, Jan/Feb 2004
33. Pottie GJ, Kaiser WJ (2000) Wireless integrated network sensors. Commun ACM 43(5):51–58
34. Shnayder V, Hempstead M, Chen BR, Allen GW, Welsh M (2004) Simulating the power consumption of large-scale sensor network applications. In: Proceedings of the 2nd international conference on embedded networked sensor systems, Baltimore, MD, USA, November 2004, pp 188–200
35. Szewczyk R, Mainwaring A, Polastre J, Anderson J, Culler D (2004) An analysis of a large scale habitat monitoring application. In: Proceedings of the 2nd international conference on embedded networked sensor systems, Baltimore, MD, USA, November 2004, pp 214–226
36. Villas LA, Boukerche A, Ramos HS, de Oliveira HABF, de Araujo RB, Loureiro AAF (2013) DRINA: a lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. IEEE Trans Comput 62(4):676–689
37. Xu Y, Heidemann J, Estrin D (2001) Geography-informed energy conservation for Ad Hoc routing. In: Proceedings of the 7th annual international conference on mobile computing and networking, Rome, Italy, July 2001, pp 70–84
38. Yao Y, Gehrke J (2002) The cougar approach to in-network query processing in sensor networks. ACM SIGMOD 31(3):9–18
39. Zoumboulakis M, Roussos G, Poulovassilis A (2004) Active rules for sensor databases. In: Proceedings of the first workshop on data management for sensor networks (DMSN'04), Toronto, Canada, September 2004

Ozgur Sanli received his B.S. (2000), M.S. (2003), and Ph.D. (2011) degrees in the Department of Computer Engineering from Middle East Technical University, Ankara, Turkey. He has been working in the Central Bank of the Republic of Turkey as an IT security specialist since 2000. His research interests include sensor networks, networking and cyber security.

Ibrahim Korpeoglu is an associate professor in the Department of Computer Engineering, Bilkent University, Ankara, Turkey. He received his M.S. (1996) and Ph.D. (2000) degrees from University of Maryland at College Park, both in Computer Science. He received his B.S. degree (summa cum laude) in Computer Engineering from Bilkent University in 1994. Since 2002, he has been a faculty member in the Department of Computer Engineering, Bilkent University. Previously, he worked at several research and development companies in the USA including Ericsson, IBM T.J. Watson Research Center, Bell Laboratories and Bell Communications Research (Bellcore). He received Bilkent University Distinguished Teaching Award in 2006 and the IBM Faculty Award in 2009. He is a member of ACM and a senior member of IEEE.



Adnan Yazici is a full professor in the Department of Computer Engineering at Middle East Technical University, Ankara, Turkey. He received his Ph.D. in Computer Science from the Department of EECS at Tulane University in the USA, in 1991. His current research interests include intelligent database systems, spatiotemporal databases, multimedia and video databases and wireless multimedia sensor networks. Prof. Dr. Yazici has published more than 180 international technical papers. He received the IBM Faculty Award in 2011 and the Parlar Foundation's Young Investigator Award in 2001. Prof. Dr. Yazici was a Conference Co-Chair of the *23rd IEEE International Conferences on Data Engineering* in 2007, the *38th Very Large Data Bases Conference* in 2012 and Program Co-Chair of the *Flexible Query Answering Systems Conference* in 2011. He is the director of the Multimedia Database Lab at METU. He is a member of ACM and a senior member of IEEE.