Knowl Inf Syst (2011) 28:579–614
DOI 10.1007/s10115-011-0380-x

REGULAR PAPER

Energy conservation in wireless sensor networks: a rule-based approach

Suan Khai Chong · Mohamed Medhat Gaber · Shonali Krishnaswamy · Seng Wai Loke

Received: 30 November 2009 / Revised: 24 October 2010 / Accepted: 12 January 2011 /
Published online: 3 February 2011

Abstract The research reported in this paper addresses the problem of energy conservation in wireless sensor networks (WSNs). It proposes concepts and techniques to extract environmental information that are useful for controlling sensor operations, in order to enable sensor nodes to conserve their energy, and consequently prolong the network lifetime. These concepts and techniques are consolidated in a generic framework we term CASE: Context Awareness in Sensing Environments framework. CASE targets energy conservation at the network level. A subset framework of CASE, we term CASE Compact, targets energy conservation at the sensor node level. In this paper, we elaborate on these two frameworks, elucidate the requirements for them to operate together within a WSN and evaluate the applications they can be applied to for energy conservation.

Keywords Context awareness · Wireless sensor networks · Association rule mining · Energy conservation

1 Introduction

The challenge of improving sensor energy consumption is a key issue in wireless sensor networks (WSNs). This is due to the fact that sensor network lifetime is directly related to operational lifetime. If sensors can operate for longer periods of time and the frequency

S. K. Chong · S. Krishnaswamy
Centre for Distributed Systems and Software Engineering, Monash University, Melbourne, Australia

M. M. Gaber
School of Computing, University of Portmouth, Portsmouth, UK

S. W. Loke
Department of Computer Science and Computer Engineering,
Latrobe University, Melbourne, Australia

S. K. Chong (✉)
Centrelink IT, Brisbane, Australia
e-mail: cskhai@gmail.com

of node failures can be reduced, the reliability and adaptability of the sensor network will consequently improve. Furthermore, in keeping with Moore's law, technological advances in sensor processing hardware significantly outpace progress in sensor battery capacity. Thus, it becomes imperative to enhance the energy conservation in sensor networks.

In this research, we focus on the challenge of maximising sensor network lifetime. In addressing this challenge, we note that one of the important factors that affect a sensor's energy consumption is the number of operations that sensors perform to gather data about their environment. For example, a sensor that frequently collects and transmits data will consume more energy than a sensor that only periodically collects and transmits data to a central processing station. We assert that the number of sensing operations can be reduced if certain sensors' future readings can be inferred from previous readings. For instance, if it is known that all sensors in the same group give low-temperature readings after a period of time (for e.g. deployed temperature sensors that allow us to assess fire situations in a building), we can choose to switch off some sensors in the group for the specified period to reduce the number of sensing operations performed. This assertion forms the basis of this research.

In this paper, we propose concepts and techniques to extract environmental information that can be used to conserve sensor energy. These concepts and techniques are consolidated in a generic framework we term the CASE (Context Awareness for Sensing Environments) framework. The CASE framework is composed of building, learning and triggering components for conserving energy in both centralised and in-network WSN configurations. CASE is further extended to build the subset framework, CASE Compact, to support autonomous and context-aware learning of triggers that control sensors efficiently in an 'in-network' data processing environment. As part of CASE Compact, we have proposed and developed a novel rule-learning algorithm, Highly Correlated Rules for Energy Conservation (HiCoRE) algorithm, that can autonomously learn and discover rules to efficiently regulate sensing operations. Our experiences in implementing CASE and CASE Compact are reported in details benefiting practitioners as well as researchers.

CASE and CASE Compact have been implemented, and the extent of energy conservation that can result by using context-driven control of sensor operations is demonstrated through experimental evaluation. Specifically, for CASE, we have evaluated its application for a generic scenario to improve generic sensor tasks and for data muling to improve the data collection process within the application. For CASE Compact, we have evaluated two applications that have applied rules learnt from using HiCoRE. These applications include (1) physical clustering [2,39], in which rules are applied to conserve energy of cluster heads and allow more efficient clustering; and (2) query processing, in which rules are applied to reduce the amount of data transmissions required for responding to user queries.

The remainder of this paper is organised as follows. Section 2 discusses existing data-driven approaches to conserve energy in WSNs, with a later emphasis on those that have studied context-awareness in WSNs. In Sects. 3 and 4, we discuss our proposed context-aware frameworks to target energy conservation in WSNs. Following this, Sect. 5 details how both frameworks can be used in a WSN. Section 6 describes the experiments and the results obtained from our evaluation of CASE and CASE Compact. Lastly, we conclude our findings and suggest future work in Sect. 7. All code listings can be found in the appendix in Sect. 8.

2 Related work

Existing data processing approaches have focused on the challenge of running sensor operations efficiently at multiple data levels. To conserve energy, these approaches have shown that

achieving a suitable energy trade-off between computation and communication operations is important. This trade-off is achieved in these approaches generally at the network data level and the node data level. In the following subsections, we discuss approaches in these two categories with respect to how they target energy conservation and subsequently how they fit into the context of our work.

2.1 Network-based approaches

Network-based approaches focus on processing of sensory data collected at a resource-rich central location (for instance, a base station) in a sensor network in order to conserve energy. This rationale has been adopted in sensor data sampling techniques, WSN monitoring applications and sensor data querying systems with respect to *data transmission control* and *network adaptation*. In these categories of approaches, the main idea is that if sensor data correlations or probabilistic models about sensor data are derived at the base station, the base station selectively choose nodes in the sensor network to send their data samples during data gathering. As a consequence, while sampling sensor nodes, the energy consumption of nodes is minimised through reducing the number of data samples that needs to be collected at the base station.

With respect to *data transmission control* techniques, spatial/temporal correlations and probabilistic models have been used as means to efficiently control data collection. In particular, spatial and temporal correlations have been utilised to more efficiently perform sampling. In [27], the blue noise sensor selection algorithm is presented to selectively activate only a subset of sensors in densely populated sensor networks while sampling. This selection of nodes is derived through building a statistical model of the sensor network nodes known as blue noise pattern (typically used in image processing applications). This model is used to deselect sensor nodes that are active while maintaining sufficient accuracy in sensing. The algorithm used also ensures that sensor nodes with the least residual energy are less likely to be selected through incorporating of an energy cost function in the sensor selection algorithm. This approach is an improvement over existing coverage-preserving approaches such as PEAS [37] and node scheduling [33] that do not consider load balancing in node selection. Probabilistic models have also been used as a means to reduce the amount of communication from sensor nodes to the network's base station [10,12,34]. In [12], this is achieved through utilising a probabilistic model based on time-varying multivariate Gaussians to derive the probability density function (pdf) over the set of sensor attributes present in a stored collection of data. The resulting probabilities can then be used to derive spatial and temporal correlations between sensing attributes, which are consequently utilised to form the answer to any user query. Their approach conserves energy as these correlations can be used to develop a query plan that chooses the most cost-effective plan to query the sensors. For instance, in answering a query, the model might suggest a plan to sample a voltage sensor (voltage reading used to predict temperature reading) rather than the temperature sensor as the cost to sample a temperature sensor is higher.

In relation to *network adaptation* techniques, a network-based approach can conserve energy through adapting sensor network operations to physical parameters in the network such as energy levels or to adapt sensing to specific requirements of the WSN monitoring application (for instance, perform sensing only when an expected event is likely to occur). Approaches described in [11,26] have explored these aspects for energy conservation. In [26], a system-level dynamic power management scheme is used, which adapts sensing and communication operations to the expected occurrence (in terms of probability measure) of the event being monitored. This approach is proposed as traditional power management schemes

only focus on hardware idleness identification to reduce energy consumption. Their results have shown a gain of 266.92% in energy savings compared to the naive approach that does not apply such adaptation behaviour. Separately, in [11], the authors propose the use of low-power sensors as tools for enhancing the operating system-based energy management. From readings obtained by low-powered sensors, the system would infer the user intent and use it to match user needs to conserve energy. For example, energy-consuming cameras that capture user faces would turn off power until low-power thermal sensors indicate a user's presence. They have evaluated this approach by an experimental setting consisting of a camera that periodically captures images and a face detection algorithm that determines the presence or absence of a user. When a user is present at the screen, the display on the computer will be turned on and vice versa. An average energy saving of 12% has been reported respectively for their prototype using low-power sensors.

2.2 Node-based approaches

Node-based approaches focus on processing sensor data on a single sensor node that have higher processing capabilities. These are approaches that extract useful information from sensor stream data on the sensor node in a resource-efficient manner. These include data-centric routing techniques used to determine the structure of communication in the WSN in an energy-efficient manner, described here as *aggregation-based topology control* and data processing techniques that have been proposed to conserve energy at the node level, described here as *data granularity control* techniques.

Some instances of aggregation-based topology control approaches include Synopsis Diffusion [25] and Tributaries and Delta [23]. In [25], a ring topology is formed when a node sends a query over the sensor network. In particular, as the query is distributed across to the nodes, the network nodes form a set of rings around the querying node (i.e. the base station). Nevertheless, although the underlying communication is broadcast, the technique only requires each node to transmit exactly once, allowing it to generate equal optimal number of messages as tree-based schemes. However, in this technique, a node may receive duplicates of the same packets from other neighbouring nodes, which can affect the aggregation result. Improving over [25] and tree-based approaches, [23] have proposed an algorithm that alternates between using a tree structure for efficiency in low packet loss situations and the ring structure for robustness in cases of high packet loss. Topology control protocols can also be clustering-based. In the clustering scheme, cluster heads are nominated to directly aggregate data from nodes within their cluster. In physical clustering, the clustering is performed on the basis of physical network parameters such as a node's residual energy. Physical clustering protocols include Hybrid Energy Efficient Distributed Clustering: HEED [39], Low Energy Adaptive Clustering Hierarchy: LEACH [17] and their variants that run on sensor nodes. In [17], the LEACH protocol proposed allows nodes distributed in a sensor network to organise themselves into sets of clusters based on a similarity measure. Among these sets of clusters, sensors would then elect themselves as cluster heads with a certain probability. The novelty of LEACH lies in the randomised rotation of high-energy cluster-head selection among sensors in its cluster to avoid energy drain on a single sensor. Finally, local data fusion is performed on cluster heads to allow only aggregated information to be transmitted to the source, thereby enhancing network lifetime.

In contrast, *data granularity control* approaches refer to those that reduce the amount of data communicated in-network and thus prolong sensor network lifetime. The specific type of approach that can be applied to reduce communication is dependent upon the granularity of data required for the WSN application. For instance, a critical-sensing WSN application such

as smart home health care systems [4] would require accurate values from sensor nodes at all times to monitor patient health conditions, whereas coarse-granularity data would suffice for certain event detection systems, for e.g. [16].

As a consequence, various data granularity control mechanisms have been proposed to cater to the data requirements of WSN applications, utilising *compression*, *approximation* and *prediction*. In relation to *compression*, studies that have manipulated the data communication structure to more efficiently perform compression include [3,28]. In [28], a scheme known as 'Coding by Ordering' has been proposed to compress sensory data by encoding information according to the arrival sequence of sensory data packets. The compression scheme merges packets routed from nodes to base station into one single large packet up an aggregation tree. The main idea used in compression is to disregard the order in which sensor data packets reach the base station in order to suppress some packets sent from intermediate aggregator nodes to the base station. In effect, this omits the encoding required for the suppressed packets and thus improves the efficiency to perform compression. Conversely, in applications where *approximations* in the collected data can be tolerated, approximations on sensory data can be performed instead to further reduce data transmissions. The application of approximation to reduce data transmissions in-network has been explored in works such as [7,40]. These studies have explored the computation of aggregates in sensor network data. In [40], the authors propose protocols to continuously compute network digests. These digests are specifically digests defined by decomposable functions such as *min*, *max*, *average* and *count*. The novelty in their computation of network digests lies in the distributed computation of the digest function at each node in the network in order to reduce communication overhead and promote load-balanced computation. The partial results obtained on the nodes are then piggybacked on neighbour-to-neighbour communication and eventually propagated up to the root node (base station) in the communication tree. The third class of techniques that can be applied to reduce network data transmissions is prediction. Prediction techniques at the node level derive spatial and temporal relationships or probabilistic models from sensory data to estimate local data readings or readings of neighbouring nodes. When sensory data of particular sensor nodes can be predicted, these sensor nodes are then suppressed from transmitting the data to save communication costs. Similar to compression and approximation, prediction-based techniques are also required to run in a light-weight manner on sensor nodes. In the literature, prediction techniques have been proposed as algorithms for enhancing data acquisition in [22] and [14] as well as generic light-weight learning techniques to reduce communication costs in transmissions.

2.3 Summary

In essence, these approaches have explored how sensor communication can be improved through utilising processed data obtained from sensors. However, these approaches have not yet explored how results obtained from computation can be used explicitly to drive energy-efficient sensor operations. In other words, the notion of using computed information (locally at sensor node or globally at application) to autonomously decide how a sensor should operate efficiently at any point during its sensing task has not been explored. For example, if the computed information suggests that sensing is no longer required at a sensed region, then the energy-efficient operation to be carried out by sensors in that sensed region is to sleep.

A data-driven technique that can provide direct control of sensor operations is necessary to further conserve energy in wireless sensor networks. From our observation of related work in data processing approaches, we discover that a subset of network-based approaches using *context* to make sensor operations more energy-efficient have research potential towards

developing this technique. The context in their work refers to meaningful information discovered in a sensor network, in which context characterises sensed information by the sensors about their environments. In their research [11,13], the authors derive contextual information about sensors to reduce the amount of sensing and communication required. For example, in [13], the authors focus on using spatio-temporal correlations in sensor data as contextual information about the network so that sensors can avoid sending information where possible, while in [11], work is presented on using low-power sensors to detect user intent in the application to save energy. Nevertheless, their work is not targeted towards energy conservation. For example, in [13], spatio-temporal correlations are used as context while context in a sensor network also covers other information such as remaining energy resource and sensed readings. Thus, this paper focuses on the problem of obtaining contextual information in a WSN to enable energy conservation.

3 CASE framework

We propose the Context-Awareness for Sensing Environments (CASE) framework to achieve context-aware management and control for energy conservation in sensor networks. The CASE framework is built upon the notion of using available discovered contextual information in a sensor network to drive sensor operations efficiently, whereby contextual information can be regarded as any information that can be used to describe the situation of a sensor. The notion of context use in a WSN was first presented in [9].

As illustrated in Fig. 1, CASE includes three main components: (1) Learning Component; (2) Context Processor; and (3) the Context Trigger Engine and supporting datastores that include (a) Rules datastore, (b) Rules-Context datastore, (c) Context-Action datastore, (d) Action-Operations datastore, and (e) Sensor Group Information datastore.

Sensor data readings that arrive at CASE are handled by the learning component and the context processor component. The learning component is responsible for discovering rules useful to define relationships between sensor readings to be used within CASE. These rules provide the basis for triggering to happen. As an example of a rule, given three sensor nodes S_1, S_2 and S_3 that detect temperature readings in a sensor network, a rule might suggest that when S_1 and S_2 have temperature readings over 30 degree celsius, sensor node S_3 would have a temperature above 30. The reason for the association between sensors S_1, S_2 and S_3 could be that they are all deployed in the same region where the situation context is summer and temperatures are high. A rule such as "$S_1 = `H' \wedge S_2 = `H' \rightarrow S_3 = `H'$" has two parts: (1) antecedents from the left hand side of the rule (i.e. S_1 and S_2 have high temperatures) and (2) consequents from the right hand side of the rule (i.e. S_3 has high temperature).

On one hand, the antecedents of the rule imply the context of the situation in which some action can be taken when CASE next determines that this context has occured again. On the other hand, the consequents of the rule provide the action to be taken. In this example, when sensors S_1 and S_2 have high temperature readings, one action that CASE can take is to sleep 'S_3' while the rule holds true. The rules would be stored in the Rules datastore to be used by that the user or a script to program the Rules-Context datastore and Context-Action datastore in CASE. The rules can be obtained by mining sensor data through the learning component or they can be supplied by the user from prolonged observation of trends in sensor data:

(1) Direct mapping from sensor readings using static rules by user The rules in the rules datastore can be manually programmed by users. The user can program the rules by observing the trends in sensor data stored over time. Manually programming

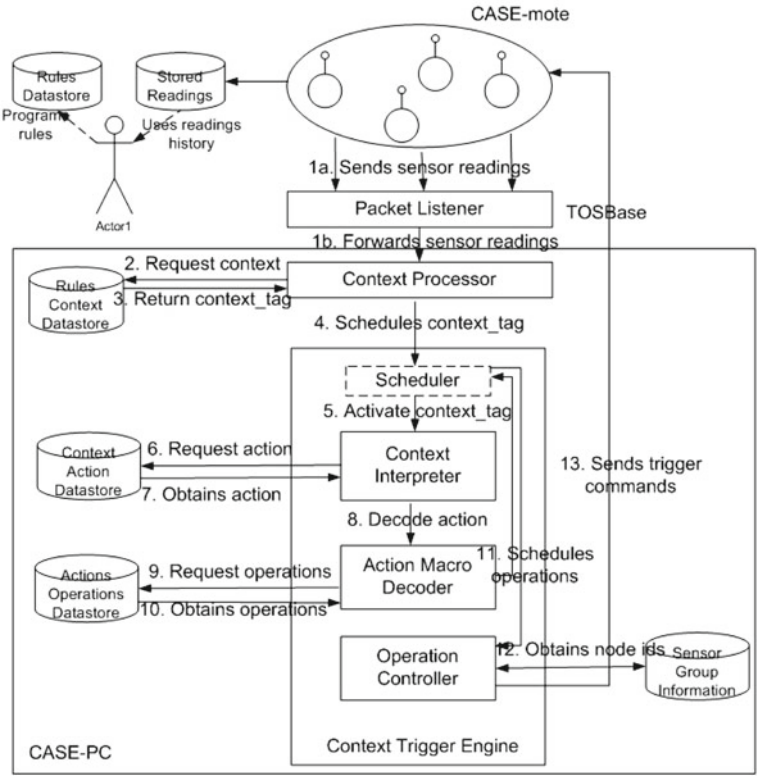


Fig. 1 CASE architecture

rules will complement any other approaches in which the rules need to be updated frequently to adjust to any sensor data distribution changes. Nevertheless, incorrect rules can result in rule conflicts or context switches and as a result, possibly increase energy consumptions.

(2) Learning Component mines for rules from sensor readings An adaptive approach to programming the rules can be provided through the use of a learning component to mine rules from sensor data. The learning component within CASE can be implemented in two ways to form rules: (1) by mining sensor data streams using existing stream association rule-mining techniques to track frequent items in sensor data streams [20] and generate rules from the frequent items and (2) by mining static sensor databases using conventional rule-mining techniques [1,29]; this requires storage of sensor readings in databases.

The datastores used by the components within CASE store information regarding contextual data, actions that can be executed and rules that describe conditions for which actions that are to be executed given the discovered context. These information can be more formally defined as events, conditions and triggers, commonly used for past active database systems [36]. Active database systems is a form of database technology that focusses on creating a system that would perform certain operations to react to expected events that can occur at runtime. The reactive nature of such a system is realised through the definitions of functions to be performed, storage for conditions to be met to trigger these functions and the expected events. As this research uses the same rule triggering paradigm as active database

systems, the following discussion conceptualises the elements stored by the CASE datastores using some data definitions adopted by active database systems.

An active database system is centred on the notion of rule, and the rule can be defined in terms of three data parts: (1) **Event:** which could results in the triggering of a rule; (2) **Condition:** which is checked in the process of rule triggering; and (3) **Action:** which is performed when the rule is triggered and the conditions have been satisfied. These declared ECA rules (Event-Condition-Action rules) can then be applied in the process of events monitoring by an active database system.

In relation to our work, an ECA rule in CASE can be formally defined as follows:

Event The event in CASE is the discovered contextual information represented by the context_tag C_i in the set C, in which $C = \{C_0, C_1, \ldots, C_k\}$ is the set of context_tags that can be discovered in the system, where $C_0, C_1, \ldots, C_k$ are distinct and $|C| > 0$. C_k is a string representation composed of numerical values (0–9) and alphabetic characters (a–z, A–Z). In other words, the context_tag is a user-defined textual representation used within the system to describe the current context for a given rule. This is given based on the user's knowledge of the application and his interpretation of the learnt rules in the Rules datastore. The context_tag is used to complement rules that describe relationships among subsets of sensors and not all sensors in the sensor network.

Condition The conditions in CASE are IF-ELSE statements that define the thresholds of sensor attribute values that need to be met to establish that any particular event has occured. The operators used for the conditional statements include $<$, $>$, $>=$, $<=$ and $==$. The threshold values that need to be met apply to selected attributes belonging to the set of all possible sensor in the WSN, $S = \{S_0, S_1, \ldots, S_m\}$, where $S_0, S_1, \ldots, S_m$ are distinct and $|S| > 0$. S_m in turn can belong to a sensor group G_o is in the set G, in which $G = \{G_0, G_1, \ldots, G_p\}$, $|G| > 0$ and $G \subset S$.

Action The action in CASE is the action_macro m_l in the set M, in which $M = \{m_0, m_1, \ldots, m_j\}$, $|M| > 0$, is the set of macros used that can be triggered in response to the event, or more specifically given any context_tag C_i. The macro m_l can eventually be reduced to operations executed by sensors.

Information regarding the events, conditions and actions can be stored textually or in any other metadata types such as XML. The relationship between context_tag, action macros and operations is illustrated in Fig. 2.

As an example, let us reconsider the rule "$S_1 = `H' \wedge S_2 = `H' \rightarrow S_3 = `H'$". The following information is obtained from this rule: given the readings of sensors S_1 and S_2, there is a high possibility that S_3 would have a similar reading. This entails that some of the sensors deployed in the same region as S_3 can reduce their sensing frequency or be put to sleep because some abnormal behaviour of the sensor is unlikely, as long as the rule holds true.

Thus, from this rule, the context_tag 'SUMMER' may be set to identify the event in which the condition 'IF both $S1$ and $S2$ having high temperature values' needs to be satisfied. If CASE detects this the event and which satisfies the conditions, the assigned action macro, REDUCE_3, is triggered. Within the Context Trigger Engine, REDUCE_3 is then decoded to the corresponding operation TRANSMIT_RATE&10000&3 to conserve energy. As a result, this particular sensor operation reduces the sampling rate to 10,000 milliseconds for sensor S_3.

3.1 CASE implementation

Implementation of the CASE framework is done on a laptop as shown in the setup in Fig. 3. In this figure, three Berkeley mica2 motes and one Berkeley mica2dot equipped with sensor

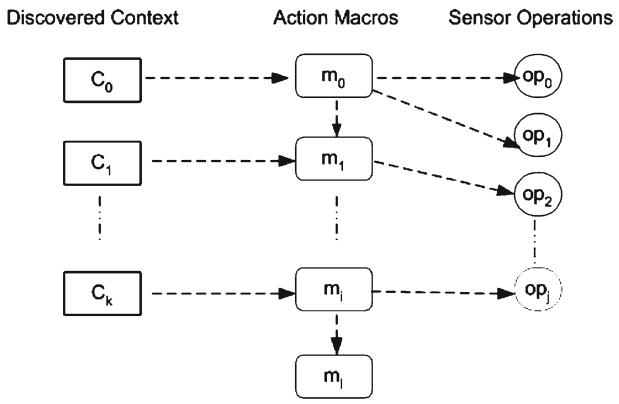


Fig. 2 Relationship between context_tags, actions and operations

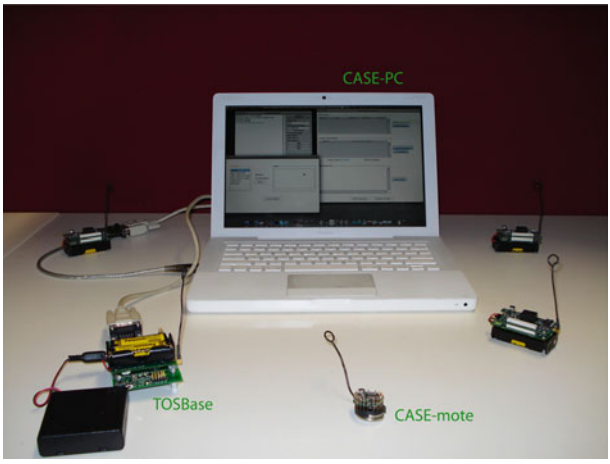


Fig. 3 CASE implementation setup

boards are connected wirelessly to the laptop through the MIB510 programming board. The Intel Core Duo laptop is loaded with the Mac OSX 10.5.4 operating system and runs Java 1.5 while the nesC implementation runs on the motes. When packets arrive from the motes on the laptop, it is first processed by the Java module that listens for sensor data packets from the programming board's serial port. The data packets are then forwarded by the listener to the CASE components to be processed.

More specifically, the CASE implementation can be divided into two parts, i.e. Java modules that run on the base station (which we term CASE-PC) and TinyOS modules that run on the motes (which we term CASE-mote, for CASE enabled motes). The Java modules provide the functionalities for the base station to receive as well as process data packets, while the TinyOS modules provide the functionalities for motes to understand triggers that are later issued by the base station. Figure 1 provides a more detailed view of the CASE framework's implementation. It depicts the main components of the CASE framework implementation and the detailed dataflows between these components. In essence, the CASE modules implemented consist of:

1. CASE-PC, which is the Java implementation of CASE modules that run on the server.

2. TOSBase, which is the TinyOS CASE module implementation for the base station node in order to provide a communication bridge between the serial and wireless channel. This implementation is provided as part of TinyOS installation and readers are referred to [5] for more detailed technical specifications regarding TOSBase.
3. CASE-mote, which is the TinyOS CASE module implementation that runs on sensor nodes, provides the functionality to communicate sensed data readings to a central base station as well as to receive and activate trigger messages from the central base station. The CASE-mote uses existing library components from TinyOS such as ADCC, Temp and Photo to support sensing capabilities, while components such as CC1000RadioIntM and HPLPowerManagementM are used to drive energy-saving functionalities.

These modules are described in greater detail in Appendix 2. The implementation of the CASE framework caters to a centralised processing environment. In this implementation, a resource-abundant base station is assumed in which sensor data readings are aggregated. In cases where sensors have to be deployed remotely without such a resource-rich base station, the use of the CASE framework is then not possible. In order for sensors to be deployed remotely without the need of a base station and adapt to context dynamically, a framework designed to work autonomously on the sensor node platform has been proposed in the form of CASE Compact. Furthermore, this entails a need to learn rules autonomously on sensor nodes in a *resource-optimised* manner, which can then be applied as adaptive triggers in remote sensing environments.

4 CASE Compact framework

As an extension of the CASE framework, we propose the CASE Compact framework to target energy conservation at the node level. Using CASE Compact, the data are processed locally at sensor nodes that collect sensed readings through their sensing capabilities and from neighbouring sensor nodes. CASE is infeasible to be used directly on sensor nodes due to its use of heavy-weight learning and triggering components. Conversely, CASE Compact uses a lightweight rule-learning algorithm that we have first presented in [8]. It is designed to operate in both homogeneous and heterogeneous WSNs.

As illustrated in Fig. 4a, CASE Compact is made up of two core components, namely the mining component and the triggering component. These two components are further described below:

(1) Mining Component: The mining component is designed to parse and learn from sensor data arriving at central nodes. This is a node centralised model in which it is assumed that the central node has relatively higher resource capabilities than the other sensor nodes that send data to it. The notion that the central node has greater resource capabilities is necessary because by running CASE Compact, it will require additional resources to mine for patterns and do triggering. This data processing model is illustrated in Fig. 4b. For a homogeneous WSN, the node M can be periodically nominated through a physical clustering protocol such as HEED [39] and LEACH [17] while for a heterogeneous WSN, node M can be either a sensor node with greater residual energy or a PDA. The output of the mining component is rules, useful for triggering in a WSN application. These rules are generated via a novel rule-mining algorithm, termed HiCoRE (Highly Correlated Rules for Energy Conservation), that we have developed to work on sensor nodes. The novelty of HiCoRE is in its ability to extract only rules for frequent transactions that contain highly correlated sensor attributes; a high correlation between

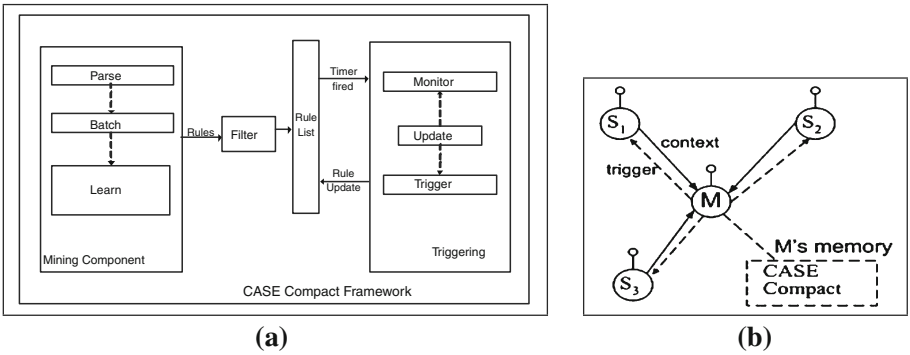


Fig. 4 CASE Compact framework and setup. **a** CASE Compact framework. **b** Homogeneous network with a central node

attributes is desirable because it signifies a strong relationship between attributed sensor nodes and in effect, increases the relevance of the rule generated. We have demonstrated the effectiveness of using HiCoRE to conserve energy in [8] (referred to as ARTS).

(2) Triggering Component: The triggering component is designed to allow central nodes to use the generated rules from the mining component to control sensor nodes in its group. Using the obtained rules with the triggering component, the central node can then send messages to power-save sensors while the rules applied infer their values. This component is useful to conserve energy for the node sensing process in applications such as query processing and event detection.

4.1 CASE Compact implementation

This implementation serves to evaluate the performance of the CASE Compact framework that we have described in chapter 4. To recapitulate, CASE Compact involves the use of a light-weight mining algorithm to mine for highly correlated rules on the sensor node in a light-weight fashion for energy conservation in WSN applications such as triggering, physical clustering and data querying. In general, in these applications, selected nodes are used to run HiCoRE on data streams coming from neighbourhood sensor nodes to yield rules that conserve energy in specific processes for these applications (for e.g. reclustering process in physical clustering). As such, to evaluate HiCoRE on the sensor node platform, the implementation of CASE Compact has been performed for a homogeneous network of Berkeley motes, similar to Fig. 4b. This setup involves three Berkeley mica2 motes and one Berkeley mica2dot equipped with sensor-boards, which communicate readings to one another wirelessly.

For the CASE Compact framework to operate in a homogeneous network, there is a need for representative nodes to be temporarily nominated in order to mine for rules from sensor data collectively and coordinate triggers. As an example, a mica2dot may be designated as the node to collect and process data coming from all 3 mica2 motes. It is noteworthy that this nominated node is yet another node in the network and does not need to be a resource-rich base station or central base station. To enable the selection of such nodes in an autonomous manner dynamically, this implementation uses the state-of-the-art physical clustering algorithm, HEED (Hybrid Energy-Efficient Distributed Clustering) [39]. HEED has been chosen for this implementation because it has shown favourable results compared to current existing physical clustering algorithms such as LEACH [17], which only selects random cluster

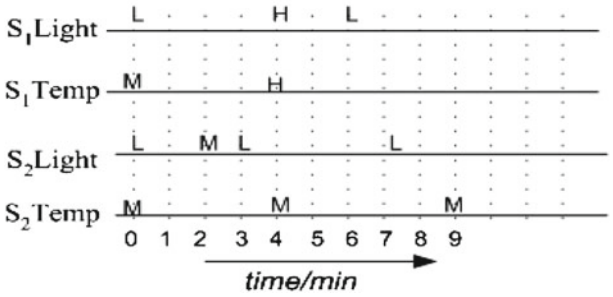


Fig. 5 Sensor data arrival

heads. Furthermore, HEED has a real sensor node implementation, for which the source code is available online [38]. This makes HEED ideal as a case study in our experiments. Details of the implementation of CASE Compact on top of HEED can be found in Appendix 3.

In this implementation, nominated HEED cluster heads are utilised to learn rules from sensor data streams that arrive from cluster members. The resulting benefit is energy conserved at the node level in a physical clustering application. As a whole, this implementation serves the core purpose of evaluating the energy conserved while using CASE Compact in an in-network processing environment. To further explain how CASE Compact operates, we provide the HiCoRE algorithm in the next section and use a homogeneous network with a head node similar to HEED as reference.

4.2 CASE Compact: HiCoRE algorithm

4.2.1 Definition

Let us consider the homogeneous network in Fig. 4b where the nominated node M is assigned to receive data readings from a group of sensor nodes. Let $S = \{S_0, S_1, \ldots, S_n\}$ be the set of sensor nodes in this group and let $A = \{a_0, a_1, \ldots, a_m\}$ be the set of attributes for any sensor $S_i \in S$, $S_i \neq S_j$ for $0 \leq i, j \leq n$, $a_x \neq a_y$ for $0 \leq x, y \leq m$. Any attribute a_x can represent sound, temperature, light or any other such sensing attributes for which the data may be gathered. Also let $S_i A = \{S_i a_0, S_i a_1, \ldots, S_i a_m\}$ denote all attribute values of any sensor $S_i \in S$. Thus, SA denotes the set of all attributes for sensor nodes in the WSN. Let $SA = \{x_0, x_1, \ldots, x_t\}$ where x_i represents any $S_i a_j$; $x_i \neq x_j$ where $0 \leq i, j \leq t$.

With reference to Fig. 5, the data sensed for any attribute a_x of S_i are continuous data that would arrive in a random manner at sensor node M. As sensor nodes are resource-constrained with limited processing capabilities, it is infeasible for sensor nodes to perform intensive computations on raw sensor data. To reduce the data processing involved to generate rules, we propose to consider only discretised sensor values in which the range can be predefined by the user/application. For example, for a light reading in [0, 1,000], the user can assign the discrete label 'L' for readings in range [0, 299], 'M' for readings in range [300, 699] and 'H' for readings in range [700, 1,000]. Similarly, for temperature readings, the representation can be 'L' for a temperature below 16 degrees, 'H' for temperature above 25 degrees and 'M' for anything in between.

Transactions are processed in batches $b_0, b_1, \ldots, b_l$, where $b_i (0 \leq i \leq l)$ must be sufficiently large so that the transactions in b_i cannot occur with equal probabilities. Let $T = \{t_0, t_1, \ldots, t_p\}$ be the set of transactions that any b_i has, where any transaction t_k, where $0 \leq k \leq p$, has the form $\{S_0 A, S_1 A, \ldots, S_n A; count(t_k)\}$ and count is the frequency in time

units (e.g. seconds) in which any attribute a_x of any sensor S_i has remain unchanged. Let $T_{\text{match}}(S_i a_x) = \{tm_0, tm_1, \ldots, tm_s\} \in T$, where $0 \leq s \leq p$ be the set of transactions in b_i that contain the attribute $S_i a_x$. Also let $T_{\text{match}}(S_i a_x = X) = \{tmv_0, tmv_1, \ldots, tmv_r\} \in T_{\text{match}}(S_i a_x)$, where $0 \leq r \leq s$ be the set of transactions in b_i having the attribute $S_i a_x$ with the value of X. For a batch b_i, the *support* of any transaction $t_k \in T$ is simply:

$$support(t_k) := \frac{count(t_k)}{\sum_{i=0}^{p} count(t_i)}. \tag{1}$$

$$support(S_i a_x = X) := \frac{\sum_{i=0}^{r} count(tmv_i)}{\sum_{j=0}^{s} count(tm_j)}. \tag{2}$$

Subsequently, the *confidence* of a rule, for instance, $(S_1 temperature = X \rightarrow S_1 light = Y)$ i.e., $(S_i a_x = X) \rightarrow (S_i a_y = Y)$, generated from a transaction over a user-defined support is given by:

$$confidence := \frac{support(S_i a_x = X, S_i a_y = Y)}{support(S_i a_x = X)}$$
$$\text{where } support(S_i a_x = X, S_i a_y = Y) := \frac{|T_{\text{match}}(S_i a_x = X)| + |T_{\text{match}}(S_i a_y = Y)|}{|T_{\text{match}}(S_i a_x)| + |T_{\text{match}}(S_i a_y)|} \tag{3}$$

The support measure can be used to describe how frequent a transaction is among all the transactions that have been collected. In the Mining Component of CASE Compact, the most frequent transaction is used to directly generate these rules. This is contrary to work by [21] in which rules are generated from all permutations of frequent itemsets from all transactions. This approach does not selectively generate rules that would be useful for energy conservation. As a consequence, their approach for rule generation cannot be directly applied for our purpose. Thus, the following sections further discuss our algorithm that selectively generates rules for energy conservation. It focuses on highly correlated sensor data attributes and only on transactions that are frequent in a batch.

4.2.2 Case Compact: mining component

The HiCoRE can then be applied to mine rules from sensor data packets arriving at node M. The rules discovered can then be used by node M to infer readings of M's neighbours and control their operations by the CASE Compact triggering component. The HiCoRE algorithm is shown as Algorithm 1.

The rules obtained by applying the algorithm can then be filtered by confidence and support. The filtered rules are then handled by the Triggering Component.

4.2.3 CASE Compact: triggering component

The Triggering Component is designed to apply generated rules from the Mining Component to enhance node sensing by energy-efficient control. This involves applying the rules generated to control the sensor nodes in a group. Consider again the network setup in Fig. 4b where sensors S_1 and S_2 communicate their readings periodically to M, and that rules have been learnt and filtered/selected using HiCoRE (Fig. 4b). To apply a rule stating that the attribute a_x of sensor S_1 implies attribute a_y of sensor S_2 (i.e. $S_1 a_x \rightarrow S_2 a_y$), the Triggering

Algorithm 1 HiCoRE Algorithm: Miner

Require: Transaction batch b_j
1: BEGIN DECLARATION
2: Set $frequentTransactionsList$ as the frequent transactions list, $frequentTransactionsCount$ list to store counts for each respective frequent transaction in the list.
3: Set $frequentTransactionsCount(t_i)$ to represent the count for any frequent transaction t_i and $mostFrequentTransaction$ denote the most frequent transaction in $frequentTransactions$, where $2 \leq |frequentTransactions| \leq t$, t is the maximum number of attributes in SA and two is the minimum required for triggering.
4: Set $maxSupport$ as the current maximum support at any time in the algorithm, $highestSupport$ as the support for the most frequent transaction in $frequentTransactions$, and $thresholdSupport$ as the user-set threshold support.
5: END DECLARATION
Ensure: Rule list R
6: Obtain energy levels of sensors in S and sort them in ascending order of energy levels, sorted energy lists $Energy_S = e_0, e_1, ..., e_q$.
7: Generate the covariance matrix, C_{matrix}.
8: Using a bitmap, initialise two sensors in S with greatest probability measure from C_{matrix} and $Energy_S$.
9: Transpose continuous transaction values in b_j to discrete values.
10: **for** $i = 1$ to $|frequentTransactions|$ **do**
11: $currentSupport = \frac{frequentTransactionsCount(t_i)}{|b_j|}$
12: **if** $currentSupport > maxSupport$ **then**
13: $maxSupport = currentSupport$
14: **end if**
15: **end for**
16: **if** $highestSupport >= thresholdSupport$ **then**
17: $R = getRules(mostFrequentTransaction)$
18: **else if** Number of bits set > 2 **then**
19: **if** all bits set **then**
20: Reset all bits to 0
21: **else**
22: Remove one bit reflecting current highest correlation in matrix
23: **end if**
24: **end if**
25: Return R.

Component enables node M to command S_2 to *power save* while the reading a_x remains true. The types of *power-save* operations that can be sent from M to S_2 include:

Trigger T1 This trigger reduces the transmission overhead, i.e. transmit only a subset of attribute values A belonging to S_i i.e. $a_x, \ldots, a_y$, where $0 \leq x, y \leq m$.

Trigger T2 This trigger sleeps and wakes sensor nodes, i.e. allowing the sensor to sleep if all its attributes can be inferred.

Trigger T3 This trigger reduces the data transmission frequency.

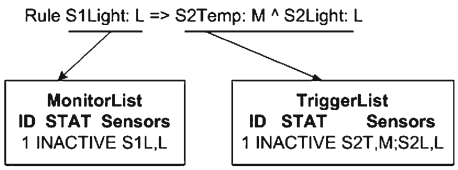
The above categories of trigger commands are not dependent on the sensor platform used. In other words, they can be applied to most sensor technologies such as Mulle, SunSpots and Berkeley Motes, in which the information for triggering can be fully provided by the rules that have been discovered. The generic process to apply the rules using the Triggering Component is as follows:

As illustrated in Fig. 6, the rule that is obtained from HiCoRE is composed of two integral parts: rule antecedents and rule consequents. Rule antecedents are the sensors and their values, from which we infer consequent sensors' values when the antecedents' monitoring values are present. To enable triggering, the monitorList is used to store antecedent sensors and their values, while consequent sensors and their values are stored in the triggerList. The

Algorithm 2 Triggering Component

Require: Rule r
Ensure: Rule Trigger
1: Divide rule into two parts with same rule id.
2: Update monitorList with antecedent sensors.
3: Update triggerList with consequent sensors.
4: Set rule to ACTIVE state for trigger on next timer fire.

Fig. 6 Trigger lists



attribute, STAT, is used to track the status of a rule trigger i.e. whether attributes should still be monitored by labelling it as ACTIVE or INACTIVE. To reference the rule being used, the values stored in both lists are tagged by an identical rule id. During operation, using these two lists, trigger messages are sent to command consequent sensors in triggerList to perform one of the trigger operations T1, T2 or T3.

5 Using CASE and CASE Compact in a WSN

This section describes how both CASE and CASE Compact can be used together in a WSN for energy conservation. In particular, it details the network characteristics that would be present when both CASE and CASE Compact operate in the same sensing environment as well as the potential benefits and drawbacks in using both frameworks together in the same WSN application.

To begin, for CASE and CASE Compact to operate in the same sensing environment, the application in which the two frameworks are used together should embody the following characteristics:

Presence of a central server: a central server with a constant power input is required to run CASE in a WSN. This server is a high-resource device such as a laptop or a PC. It should act as a central repository for sensed data so that contextual input from sensors can be processed and contextual patterns be learnt. The central server should also be able to send trigger messages to sensor nodes within its radio range.
Presence of central processing nodes: CASE Compact can be used in a sensor network either in a heterogeneous network with a static setup consisting of higher resource nodes and regular sensor nodes or in a homogeneous network with a dynamic setup by using a mechanism to subgroup nodes in the sensor network and create temporary in-network group head nodes to control their respective subgroupings.
Utilisation of a communication hierarchy: a sensor network using both CASE and CASE Compact follows the communication hierarchy as illustrated in Fig. 7a. The hierarchy depicts regular sensing nodes that are responsible for sending real sensory values to the intermediate nodes which run CASE Compact and the results from these intermediate nodes are approximated values to be forwarded to the server that runs CASE.
Coarse-granularity sensing applications: coarse-granularity applications are targeted for WSNs running both CASE and CASE Compact. Some applications that we have discussed

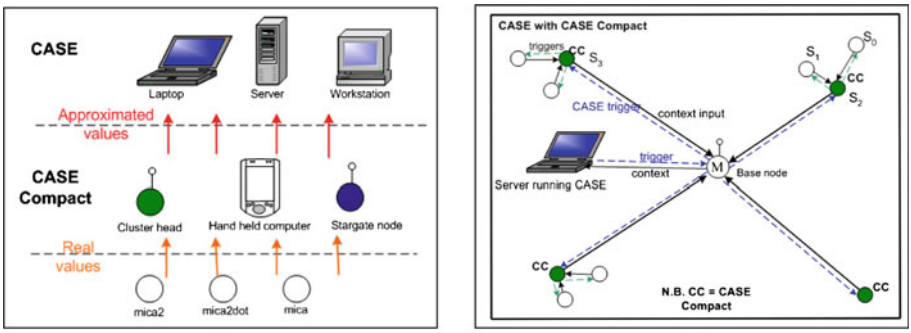


Fig. 7 CASE and CASE Compact operating environment. **a** Communication Hierarchy. **b** CASE and CASE Compact in a WSN

include event detection applications [15,16,32] and data aggregation [19]. Only coarse-granularity applications are applicable because sensor nodes running CASE Compact can only produce approximate values when triggering is used.

A WSN application that embodies the aforementioned characteristics can operate in the network arrangement that is shown in Fig. 7b. In this arrangement, CASE Compact runs on selected central nodes (green dots), while CASE runs on the server connected serially to the base node M. The central nodes that run CASE Compact is responsible for energy conservation at the node level for local nodes under their control, while the server the runs CASE is responsible for energy conservation at the group level for the sensor groups formed. For instance, in Fig. 7b, sensor nodes S_0, S_1 and S_2 belong to the single sensor group G_1. CASE controls the sensor group G_1, while S_2 running CASE Compact controls nodes S_0 and S_1. The triggers coming from CASE can apply to any of the sensor groupings that have been formed in this network.

To enable power savings in this setup, two requirements are imposed on the central node. First, the central node needs to run CASE Compact and in doing so, execute the HiCoRE algorithm to process real values coming from the nodes in its group and perform triggering to power-save-controlled nodes. The approximated values collected by any central nodes are then forwarded to the server, enroute node M. Secondly, a central node needs to respond to any trigger messages coming from the CASE server. Specifically, this response depends on the type of trigger commands the central node has received from the node M:

(1) Sleep trigger: if the trigger message is to sleep the central node and any other nodes that it is controlling, the central node would first need to sleep the nodes under its control before sleeping itself on a timer finally. When the central node has awaken, it would first broadcast wakeup messages to the sensor nodes that it has slept previously and reinitiate transmissions from those nodes. The transmission to node M from the central node then resumes to its default operation.

(2) Reduce activity trigger: the trigger message to reduce the activity level of a sensor group entails either lowering the transmission frequency or reducing the load of data transmissions. In both cases, reducing the activity level would entail the central node sending messages to the nodes in its group to inform the nodes of the intended transmission frequency or load (i.e. the message packet structures to send next) while the central node itself would send data less frequently or less data to node M.

(3) Default sensing trigger: the process to revert sensor groups to default operation follows the same process that described to reduce the sensing activity. This resumes the central node and its members to the default state programmed.

It can be anticipated that using both CASE and CASE Compact in this WSN arrangement allows more energy to be conserved, relative to only using either frameworks individually. This is because utilising both CASE and CASE Compact enables energy conservation at both node levels and application levels. At the sensor node level, patterns are learnt locally at central node to coordinate sensor operations in local groups and at the application level, more general network behaviours can be captured to drive operations for larger groups of sensors.

6 Evaluation

In this section, we evaluate the performance of CASE and CASE Compact implementations where the results obtained are presented and analysed. The aim of the experiments conducted with CASE is to evaluate energy conserved through using discovered contextual information at the network level to control sensor node operations. This objective has been achieved through using our CASE implementation in a pig sty scenario. It needs to be noted that though these results have appeared in [9] and [8], this is the only work that has given a complete explanation of the integration of the two frameworks and has expressed our results in greater detail and in that context.

6.1 CASE evaluation

The aim of the experiments conducted with CASE is to measure the energy saved when CASE has been utilised to drive sensor node operations in a pig sty scenario. It needs to be noted that the purpose of the experiments performed is to demonstrate that energy can be conserved (through applying triggers such as sampling rate adjustment) when a WSN system adapts to various types of context and not simply to show the effects of varying sampling rates on a WSN system. Thus, we have not presented any formal procedures for determining the exact sampling rate required but rather this has been left to the application designer.

In this scenario, we envisage a pig sty with deployed mica2 sensor nodes to collect sensory information about the living environment of pigs. This sensory information includes light, temperature and voltage readings. As illustrated in Fig. 8, the setup involves deploying a static node S_1 within the pig sty, a mobile node S_4 attached to pig A and a base node S_0 attached to the base station. It has been assumed that both S_1 and S_4 are within communication range of S_0. The objective is to use contextual information discovered to enable the energy-efficient control of sensor S_1 installed inside the pig sty. The contextual information in this case is primarily derived from data readings communicated to the base station by sensor nodes S_0, S_1 and S_4. More specifically, the contextual information pertains to the movement/location of the pigs.

In these experiments, we consider the following context_tags used within CASE:

1. Normal. This represents the default context of the sensors.
2. Pigs_Out. This represents the context whereby pig A with sensor S_4 attached is outside the pig sty.
3. Pigs_In. This represents the context whereby pig A is inside the pig sty.
4. Daytime. This represents the context that it is daytime and pig A is outside the pig sty for an extended period of time.

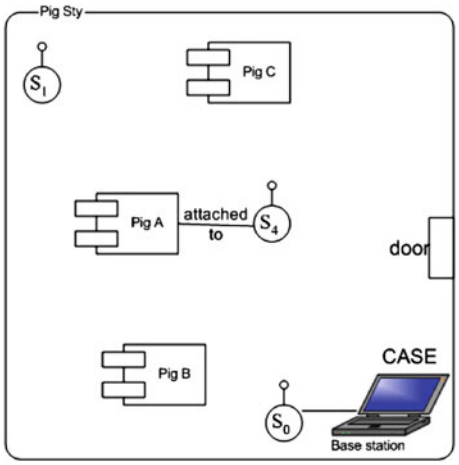


Fig. 8 Pig sty scenario

5. Nighttime. This represents the context that pig A will be inside for an extended period of time.

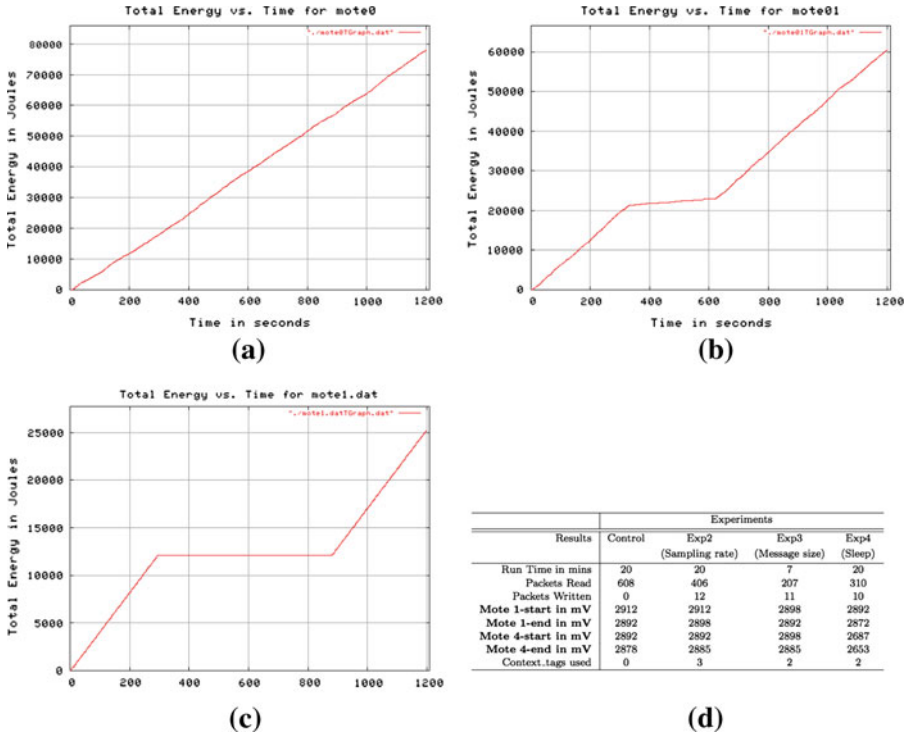
Results from the experiments performed using various types of triggers are recorded from actual runs (given in Fig. 9d) as well as simulations using PowerTOSSIM [31] for each experiment individually. The PowerTOSSIM results are shown in Fig. 9. The purpose of the actual runs is to collect results pertaining to the total number of packets sent and received at the base station, whereas the simulations serve to measure the energy consumption of the sensor node that has been contextually controlled by CASE (in this case, sensor S_1). The data collected from running PowerTOSSIM are filtered using Perl scripts and presented as a graph using gnuplot. These experiments and the analysis of the results obtained are further discussed as follows:

6.1.1 Experiment 1: control experiment

This experiment serves to measure the energy consumption of a sensor node when CASE is not utilised to drive sensing operations. This experiment is used as a reference comparison in terms of energy savings for subsequent experiments. In this control experiment, sensor nodes S_0 (mote 0), S_1 (mote 1)and S_4 (mote 4) are placed in room temperature and under moderate lighting conditions. As CASE is not running, no context changes are detected to occur over a duration of the 20-min run. Motes 0, 1 and 4 collect light, temperature and battery voltage readings at every 1,000 ms and transmit the collected data packets to the base station. The default context_tag in CASE is 'Normal'. In total, 608 data packets were sent out to the base station by all sensor nodes. Figure 9a shows the cumulative energy consumption of sensor node S_0 over time. Sensor nodes S_1 and S_4 exhibit the same behaviour with respect to energy consumption. As a whole, the graph result demonstrates that the energy consumption of a sensor node is directly proportional to the duration of runtime when no other context_tags have been detected in this experiment.

6.1.2 Experiment 2: sampling rates experiment

In experiment 2, CASE is used to control sensor node S_1 by dynamically adjusting its sampling rate upon detection of a suitable context_tag. In this experiment, we first assume that



	Experiments			
Results	Control	Exp2 (Sampling rate)	Exp3 (Message size)	Exp4 (Sleep)
Run Time in mins	20	20	7	20
Packets Read	608	406	207	310
Packets Written	0	12	11	10
Mote 1-start in mV	2912	2912	2898	2892
Mote 1-end in mV	2892	2898	2892	2872
Mote 4-start in mV	2892	2892	2898	2687
Mote 4-end in mV	2878	2885	2885	2653
Context_tags used	0	3	2	2

(d)

Fig. 9 Pig sty experimental results **a** Control experiment: overall energy versus time. **b** Sampling rates experiment: overall energy versus time. **c** Sleep node experiment: overall energy versus time. **d** Table of experimental results

S_1 can sample at three different rates within CASE, i.e. at 1,000 ms, at 10,000 ms and at 100,000 ms. Let us also assume that the current context_tag in CASE is 'Pigs_In' and S_1 is sampling at the rate of 1,000 ms. When pig A with S_4 attached moves out of the sty, high light readings detected by S_4 outside enables the detection of the context_tag 'Pigs_Out' by CASE. As a consequence, CASE sends the trigger to reduce the sensing rate of S_1 in the sty to 10,000 ms. Subsequently, the sensing rate of S_1 is returned to 1,000 ms when the pigs return to sty and the context_tag is 'Pigs_In'. The results shown in Fig. 9b illustrate the energy consumed by S_1 during the course of triggering by CASE to react to the current context_tag detected. In both figures, this correspond to the context_tag 'Normal' during the time period from between 0 and 360 s, the context_tag 'Pigs_Out' between 360 and 640 s, and context_tag 'Pigs_In' between 640 and 1,200 s.

A total of 406 data messages were sent out from all sensor nodes in this experiment. In empirical terms, the results obtained show a reduction in the total number of messages sent in the network of 33.2% ((608−406)/608)*100 when compared to the control experiment, with 12 packets written out to node S_1 from the base station to change its sampling rate. It also demonstrates an energy saving of around 22.9% (77,800−60,000)/77,800 when we compare the final energy consumed at the time = 1,200 s in Fig. 9a (i.e. 77,800) and b (i.e. 60,000). Despite the energy conserved, it can be noted that some additional energy is expended when S_1 needs to readjust its operations to the trigger resulting from a new context_tag, also termed *context switch*. These context switches occur at times 340 and 580 s, incurring time lags of

20 and 60 s, respectively. From this observation, it is likely that if context switches occur too frequently, a significant amount of energy may be expended. A possible solution is to set appropriate thresholds within CASE to limit the number of context switches, which is a possible direction for future work.

6.1.3 Experiment 3: message size experiment

This experiment serves to measure possible energy savings from using triggers in CASE to reduce the size of data messages transmitted by sensor nodes. In this experiment, we assume that two possible message structures may be set, i.e. a 24-byte data packet that encapsulates light and temperature data (packet type A) versus a 16-byte data packet that excludes the light data (packet type B). Let us also assume that the current context_tag is 'Daytime' and all light readings currently need to be recorded. However, when the lights inside the sty are switched off (at night) resulting in the context_tag 'Nighttime', light readings are then no longer required at S_1. This results in the trigger by CASE to temporarily disable light readings, i.e. the preference to choose the packet type B over the packet type A.

Figure 9d shows the results from our experimental run of 7 min when this situation happens. As message size changes are not observed in PowerTOSSIM simulations, corresponding graphs are not presented. Nevertheless, the results from the actual runs may be observed. In this run, the context_tag detected for the first 2 min is 'Daytime', switching to 'Nighttime' in the next 3 min and back to 'Daytime' until the end of the experiment. During 'Nighttime', S_1 transmits data packets without light readings at the rate of 1,000 ms to the base station. In total, 207 packets were received from sensor nodes at the base station during this period. Out of the 207 packets recorded, 41 packets had the packet type in listing 6.2 of 16 bytes and 166 have the packet type in listing 6.1 of 24 bytes. This equates to a total of 4,640 bytes received at the base station using this trigger. In comparison, if all 207 packets sent were of the packet type in listing 6.1 of 24 bytes, 4968 bytes would have been sent from the sensor nodes, implying an additional 328 (4,968–4,640) bytes which is not required, based on the current contextual information. This achieves a 6.6% reduction in data transmitted. This is a conservative estimate as the payload only omits the light readings. It is anticipated that more data transmissions can be reduced if more elements of the readings are omitted, for instance, both light and temperature readings. Furthermore, this is with respect to one sensor. A typical WSN consists of a large number of sensor nodes, to the scale of thousands. Finally, it must also be considered that that the time duration of the experimental runs is lesser than a real-world deployment where sensors would continuously monitor for several days. The implications for such scenarios clearly indicate that the CASE approach can have significant benefits.

6.1.4 Experiment 4: sleep node experiment

This experiment serves to measure energy savings from applying triggers to sleep sensor nodes by CASE. This involves putting sensor nodes to sleep over an extended periods of time to conserve energy when it is expected that sensing is not required. The adapted scenario is similar to that of the message size experiment in which the context_tags 'Daytime' and 'Nighttime' are used. Contrary to that experiment, however, S_1 is put to put sleep instead to further conserve energy until readings from S_4 indicate that it is 'Daytime' again. Figure 9c shows the energy consumption of node S_1 during the context switches, i.e. 'Daytime' between 0 and 240 s, 'Nighttime' between 250 and 630 s and 'Daytime' again from 640 s onwards. Figure 9c shows the cumulative energy consumed in this period. In comparison with the

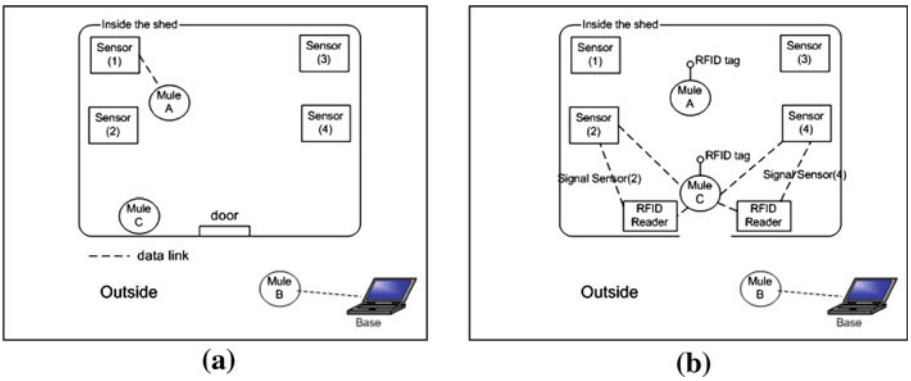


Fig. 10 Muling scenarios. **a** A basic data muling approach. **b** Muling approach with CASE

control experiment, approximately 67.9% (77,800−25,000)/77,800 is conserved when we compare the final energy consumed in Fig. 9a (i.e. 77,800) and c (i.e. 25,000). Although this experiment demonstrates that significant amounts of energy are saved when the trigger is to sleep a node, the resulting impact of the trigger to sensing operations within the WSN needs to be considered. First, a slept sensor node would be completely unresponsive for some defined time period. This is not feasible when the sensing task is mission-critical. Secondly, a sensor node that has been put to sleep needs to be awakened either through an internal timer or by a subsequent trigger due to another context_tag. In the latter case, the onus is on CASE to ensure that the sensor node put on sleep will be awakened.

6.1.5 Data muling case study

Data muling [30] typically involves the use of mobile entities known as data mules to gather data from statically deployed nodes in a network and deliver the gathered data to a base station. Data mules are used in sparse sensor networks for minimising costly data transmissions from nodes directly to base station and for balancing the load on intermediate nodes that commonly relay sensor data over long distances. Some example for real-world applications of data muling includes using nodes mounted on gardening spades as data mules in a vineyard [6] and using an AUV as a data mule to collect readings from underwater sensors [35]. In order to evaluate the energy benefits in using CASE for data muling, we propose to compare the two approaches to muling as below:

(1) *A basic data muling approach.* In this approach, static sensor nodes have to poll continuously for the mule in order to establish a connection. This approach is developed for the experimental pig facility [24] to measure the effect of external stressors in a shed (e.g. temperature) on the core body temperature of pigs. As illustrated in Fig. 10a, this involves the use of data mules A, B, C in the shed, whereby the mule can be any mica2dot mounted on a pig. The main purpose of a mule is for collecting data from the static nodes in the network (shown as sensors 1, 2, 3 and 4 in Fig. 10a) and uploading collected data to the base station outside the shed. The data communication between mule and static nodes or mule and base station occurs when both entities are in close proximity. Specifically, the data mules will collect data from the static sensor nodes when they are inside the shed and uploads the collected data to the base station when they are outside the shed.

(2) *Muling approach with CASE.* For the same scenario, we now model the static sensors (to be data muled) as "the sensors that are to be controlled by context triggers" and additional sensors on the mule (i.e. the pig) as "sensors that can provide the contextual information for control". Other sensors in the farm can provide both contextual information and be controlled. Illustrated in Fig. 10b, this setup includes sensors 1, 2, 3 and 4 as the static sensors that collect data (Berkeley motes), mules A, B and C as the data mules (Berkeley motes with RFID tags) and RFID readers as the static sensors that provide only location information to CASE on the laptop. CASE uses the location context provided by the RFID readers to trigger mule detection. In this scenario, with RFID tags attached to the data mules, once mule C enters the shed, the RFID readers located at the entrance of the shed detect mule C on entry and send the detection information to CASE. Acknowledgements are then sent from CASE to sensor 2 in order to initiate and establish a data transfer connection with mule C. Basically, sensor 2 only sends logged readings to mule C upon a trigger sent from the base station. When mule C is in safe distance of sensor 2, mule C receives data from sensor 2 and sends acknowledgements to sensor 2. Mule C remains in listening mode even if there are no more packets from sensor 2. When mule C leaves the shed again, CASE sends a trigger to sensor 2, ceasing its current communication with sensor 2. This approach improves over the basic mulling approach because the sensors are no longer required to poll for the mule but is explicitly informed when they are near enough by CASE.

The implementation in both scenarios involves using an a/c-powered laptop as the base station, mica2dots as the data mules and mica2s as the statically deployed sensor nodes that regularly sample temperature and light readings in this environment. The motes are programmed in nesC under the TinyOS[18] operating system to perform muling, while the SerialForwarder application is used to provide the required serial interface between motes and the base station. CASE runs on the base station.

The experiments conducted evaluate the performance aspect with regard to the process of initially establishing a connection between the mule and sensor/base station. In order to evaluate this performance aspect, the parameters measured in this experiment include the minimal distance between a mule and a sensor when a packet from a sensor can successfully be heard by a mule, the resulting number of packets sent from the sensor and received by the mule while they are connected, the first recorded safe distance when there is 1 or less packets lost during transmission.

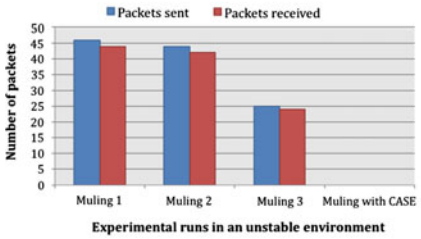
Both experiments follow the same procedures as follows. Let us assume there is a static sensor such that the mule moves in the direction of the sensor. As the mule approaches the sensor, the distance of the mule from the sensor when it is first detected by the sensor is recorded. After the mule is first detected, the connection maybe unstable until the mule is closer to the sensor. The number of packets that have been received by the mule and sent by the sensor during this period from the first detection is then recorded next. As the mule moves closer to the sensor, packet loss between the two eventually reduces to only 1 or less (i.e., when readings stabilise and the mule synchronises with the sensor). The minimal distance between the mule and the sensor when their readings stabilise (i.e. safe distance) is then recorded with the packets that have been sent and received. In terms of using CASE, the experiment is carried out in the same way but with the exception of a preset safe distance of 78 cm where RFID readers are placed. This safe distance is set on basis of results obtained from the experimental runs in Fig. 11a which have shown that only one packet would be lost when the safe distance is 88 cm or less. CASE is then expected to trigger the communication between the mule and sensor. The results obtained have been illustrated in Fig. 11b, c and d.

	Basic Muling experiments		
Experiment runs	1	2	3
Distance in cm, 1st detection	143	157	153
Packets Sent, unstable	46	46	25
Packets Received, unstable	44	42	24
Safe Distance in cm	88	105	99
Packets Sent, stable	41	48	38
Packets Received, stable	40	48	38

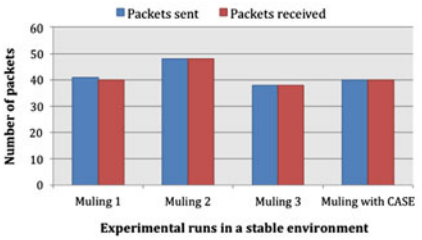
(a)

	Context-Aware experiment
Result	Context-Aware
Distance in cm, mule detected	78
Packets Sent, stable	40
Packets Received, stable	40

(b)

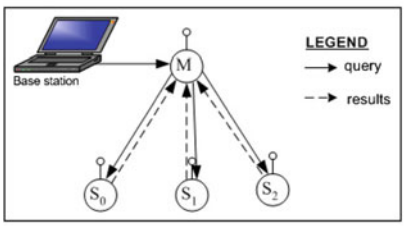


(c)

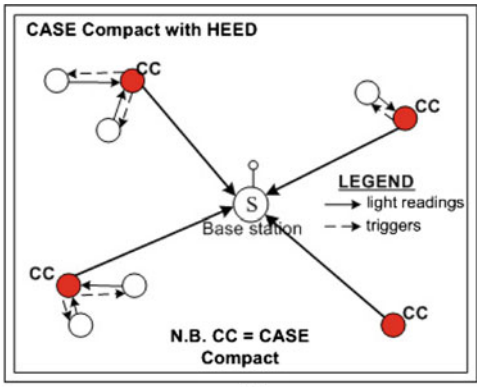


(d)

Fig. 11 Experiment results using CASE. **a** Experimental results using polling only. **b** Experimental results using CASE. **c** Overall results for transmissions in unstable environment. **d** Overall results for transmissions in stable environment



(a)



(b)

Fig. 12 CASE Compact experimental scenarios. **a** Query processing setup. **b** Physical clustering setup

6.2 CASE Compact evaluation

The aim of the experiments conducted in this section is to measure the amount of energy saved when CASE Compact is used with a query processing application(such as TinyDB) and a physical clustering application(such as HEED). First, we describe the experiments that measure the energy that can be conserved in a scenario involving the querying of a node, as shown in Fig. 12a.

As shown in Fig. 12a, this setup involves the base node M connected to a laptop and other sensor nodes S_0, S_1 and S_2 that transmit light, temperature and microphone readings to node M. In this setup, let us assume that the user creates the query at the base station. In the process of answering the user query, node M requests for sensory readings from sensor nodes

S_0, S_1 and S_2. Upon hearing data requests from node M, the sensor nodes then transmit their sensory readings to M, en route to the base station. The transmissions from sensor nodes to M continue for the lifetime of the query.

For the purpose of this evaluation, let us now consider that node M runs the HiCoRE algorithm and has obtained rules relating to S_0, S_1 and S_2 prior to running a query. The obtained rules can then be used by M to only request sensory readings from sensor nodes when the readings cannot be inferred by filtered rules, for instance, rules with high confidence values. In theory, this should reduce the amount of data communications necessary to answer a query and thus conserve energy. In order to demonstrate that energy can be conserved, the experiments that follow study how much data transmissions can be reduced in this manner and the compromise on the data quality obtained, if any, through measuring the accuracy of the rules obtained. In order to measure rule accuracy and the actual number of bytes transmitted, computer simulations are performed. This involves running a Perl script to determine the amount of data transmitted in this scenario for the lifetime of a hypothetical query when our C implementation HiCoRE is used/not used. For the run with HiCoRE, synthetic sensor data transactions are first piped to the HiCoRE program so that rules may be learnt at prior. The synthetic dataset used has the following properties:

1. S_0 light readings and S_1 light readings have a positive correlation of 0.8 ± 0.04.
2. S_1 light readings and S_1 temperature readings have a positive correlation of 0.8 ± 0.04.
3. S_2 light readings and S_2 temperature readings have a negative correlation of -0.8 ± 0.04.

This dataset is run on HiCoRE for a period of 8 min. It is noteworthy that this evaluation is performed on a synthetic dataset as this is the way we can establish the accuracy of HiCoRE by testing whether the rules predicted capture the known correlations. After 8 min, any rules generated by HiCoRE are stored in M's memory. Before node M requests sensory readings from sensors S_0, S_1 and S_2, it will first look at all the rules that have been stored in memory. If a rule in memory meets the confidence threshold, then the rule will be used. Specifically, M will use the inferred consequent values of the rule rather than requesting the necessary values from the sensors.

For both simulation runs, the query used is "SELECT * FROM sensors, SAMPLE PERIOD 1s FOR 7 min". The results obtained from running this query in the simulations are presented and analysed in the following sections, i.e. on the accuracy of rules generated and on the resulting data throughput for query with/without rule adaptation.

6.2.1 Measuring accuracy of rules

Table 1 shows the rules obtained when HiCoRE is run on the aforementioned synthetic dataset for 8 min with a set confidence threshold of 0.5. It also shows the resulting confidence of the rule generated. In order to measure the accuracy of the rules generated, each of the rule is then applied to perform prediction on the same dataset used for the query evaluation. In this regard, a prediction is considered to be successful when the predicted value is the same as the expected value in the dataset in discrete form. As an example, given the rule R1, '$S_1L[H] \rightarrow S_0L[H]$', if the light value of S_0 is correctly predicted to be low when S_1 has a high value, then the prediction is successful. For rule R1, the prediction is successful 34 times out of 41 times when S_1 has a high light reading, giving a success rate of approximately 82.9%.

Two observations can be made from the accuracy results shown:

1. The rules generated by HiCoRE are able to capture the correlations between data attributes, if they are present in the dataset. This is demonstrated through the results showing

Table 1 Table of rules discovered by HiCoRE

Discovered rules			
RuleID	Rules	Confidence	Success rate in %
R1	$S_1L[H] \rightarrow S_0L[H]$	0.83	82.9% (34/41)
R2	$S_0L[H] \rightarrow S_1L[H]$	1.0	100% (34/34)
R3	$S_1L[H] \rightarrow S_0L[H]$	0.83	82.9%
R4	$S_0L[H] \rightarrow S_1L[H]$	1.0	100%
R5	$S_0M[Y] \rightarrow S_0L[L]$	0.5	45.3% (24/53)
R6	$S_0L[L] \rightarrow S_0M[Y]$	1.0	85.7% (24/28)
R7	$S_1T[L] \rightarrow S_0L[H]$	1.0	100% (20/20)
R8	$S_0L[H] \rightarrow S_1T[L]$	0.73	58.9% (20/34)
R9	$S_1T[L] \rightarrow S_0L[H]$	1.0	100%
R10	$S_0L[H] \rightarrow S_1T[L]$	0.64	58.9%

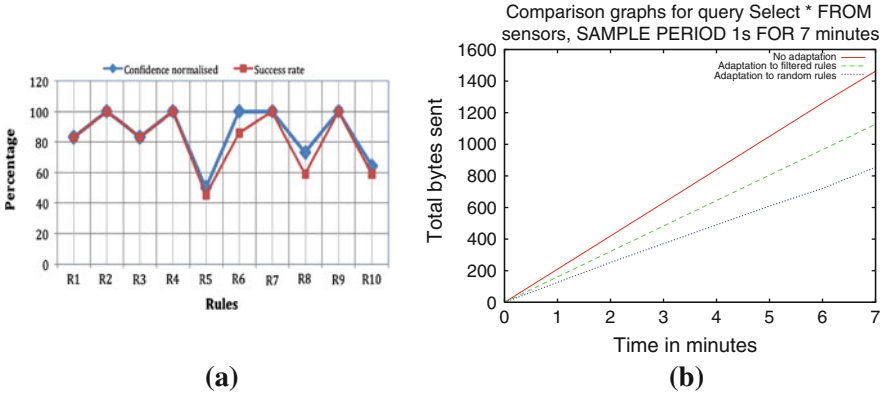


Fig. 13 Query processing application results. **a** Rule confidence and resulting success rate. **b** Comparison graphs for query Select * FROM sensors, SAMPLE PERIOD 1s FOR 7 min

that a majority of rules generated reflect the relationship between the light readings of S_0 and S_1 as well as between the light readings of S_1 and temperature readings of S_1.

2. Rules with high confidence values generated by HiCoRE typically imply a high success rate in prediction. This is further illustrated in Fig. 13a, whereby rules such as R5 having a confidence value of 0.5 (normalised to 50% in Fig. 13a) have a resulting success rate of 45.3%. Thus, confidence is a good measure of the predictive accuracy of the rule for data querying application.

6.2.2 Measuring data transmitted

Figure 13b illustrates the amount of data transmitted in bytes by sensors to M when the query is answered in three different situations:

1. No rules are used. In this situation, M requests sensors S_0, S_1 and S_2 to send their readings to M every second for 7 min.
2. Filtered rules are used. In this situation, M selects high confidence rules to be used in answering the query. For the purpose of this evaluation, let us assume that rules R6 and

R7 are used by *M* due to the fact that they have high confidence values, i.e. 1.0, and that they are not in conflict with one another, i.e. the consequent of R6 rule is not the antecedent of rule R7 and vice versa.

3. Randomly selected rules. In this situation, *M* just randomly selects two rules to be used from all the rules stored in memory. Let us assume that, in the worst case scenario, low confidence rules R5 and R8 have been selected.

From the results obtained in Fig. 13b, it can be observed that energy is conserved when rules generated by HiCoRE have been used to answer the query, regardless of whether random rules or filtered rules have been used. This is evidenced by a reduction in the overall amount of data communicated which directly impacts on the overall energy consumption by the sensor network [31]. Specifically, the adaptation to filtered rules reduces data transmission by 22.7% (1,450−1,120)/1,450, while the adaptation to random rules reduces data transmission by 41.4% (1,450−850)/1,450. While it may appear that using random rules is better than using rules filtered/selected using confidence levels, it needs to be highlighted that accuracy is reduced with random rules. In particular, it needs to be noted that the resulting error rates from prediction using random rules are much higher than using filtered rules. Empirically, when filtered rules R6 and R7 are used, the combined success rate is 91.7% (i.e. referring to Table 1, total combined errors of 4 divided by total combined results predicted of 48 from using rules R6 and R7) while when random rules R5 and R8 are used, the combined success rate is only 50.6% (i.e. referring to Table 1, total combined errors of 43 divided by total combined results predicted of 87 (53+34) from using rules R5 and R8).

In general, the results obtained demonstrate that the data transmissions required to answer a query is reduced (and in effect, energy is conserved) when CASE Compact is used in data querying. The best strategy to select rules for use within a query application is not necessarily selecting rules that conserve the most energy, but rather using a strategy that can achieve the best compromise among criteria such as the overall energy conserved, error rate that can be tolerated by the application (for instance, if the application is mission-critical, then the success rate/confidence threshold should be high) and appropriateness to the current query (for instance, using rules relating to sensors that are relevant to the query).

6.2.3 Physical clustering experiments

The experiments performed here measure the overall energy that is conserved in that scenario and also the performance of the HiCoRE algorithm operating on HEED cluster heads with respect to its overhead and scalability. The experimental setup is illustrated in Fig. 12b.

As shown in Fig. 12b, this setup involves running CASE Compact on HEED nodes. Specifically, the cluster heads formed when HEED is run (represented by red coloured nodes in Fig. 12b) are made to run HiCoRE in conjunction to learn rules about cluster members. These rules are then applied as triggers by the cluster heads to control cluster members' operations and thereby conserve energy. The trigger used in our experiments is to command a node not to send its light information.

The experiments are performed using TOSSIM simulations of the HEED implementation with HiCoRE. Each experiment runs with individual simulation settings depending on the performance aspect evaluated. The common unit of measurement used to measure overall network energy consumption is the Credit-Point System (CREP), as used by HEED in [39]. This system measures the overall network residual energy of sensor nodes by weighing the cost of the total number of intra-cluster and inter-cluster communications performed during

runtime. The resultant cost value provides an estimate to the total energy consumed for the sensor network.

With respect to Fig. 14c, the experiment measures the overall energy conserved when HEED is used with CASE Compact (CC in graphs) for different confidence levels. The simulations for this experiment are performed using 50 nodes, a seed value of 124,755 for a duration of 25 min. The simulations are performed to determine the residual energy of the network when:

(1) HEED is run alone.
(2) HEED is run with CASE Compact, whereby generated rules having a confidence value above 0.5 are used for triggering.
(3) HEED is run with CASE Compact, whereby generated rules having a confidence value above 0.7 are used for triggering.
(4) HEED is run with CASE Compact, whereby generated rules having a confidence value above 0.9 are used for triggering.

In general, the results show that HEED cluster heads running CASE Compact outperforms the nodes that run with HEED only with respect to energy conservation. For a more specific comparison, let us consider the residual energy for all runs when the simulation time is 140. At this time, the residual energies are 14,833, 17,167, 17,500 and 16,500 ($\times 10^3$) for runs **(1)**, **(2)**, **(3)** and **(4)**, respectively. This corresponds to energy savings of 15.7, 18.0 and 11.2% relative to the HEED run when the confidence threshold is 0.5, 0.7 and 0.9, respectively. However, given the trend observed in Fig. 14c i.e. the graph showing the use of HEED and the graphs with CASE Compact diverges as time passes, it can be expected that for a long enough time, further significant energy savings can be achieved.

In contrast, Fig. 14a shows the energy conserved when HiCoRE is run with a higher number of nodes. This is for the purpose of testing the scalability of HiCoRE. The simulations for this experiment are performed using 100 nodes, a seed value of 124,755 and a confidence value of 0.75 for a duration of 25 min. Comparing the results from Fig. 14a, c, it can be observed that as the number of nodes used increases from 50 to 100, there is a corresponding linear increase in the amount of energy conserved. The results also demonstrate that it is feasible to implement HiCoRE for higher numbers of nodes.

Figure 14b illustrates the energy overhead of the system when HiCoRE is run with HEED (no triggering applied) and when HEED is run alone. The simulations for this experiment are performed using 50 nodes, a seed value of 124,755 and a confidence value of 0.75 for a duration of 25 min. The results show that HiCoRE runs with marginal energy overhead when compared to the energy overhead to run HEED. To measure this explicitly, let us consider the respective residual energies at time = 140. At this time, the residual energies are 1,750 and 1,850 ($\times 10^3$) for HEED and HEED with HiCoRE, respectively. This yields an additional overhead of 0.06% (100/1,750) for HiCoRE. This overhead is very small relative to the amount of energy that can be conserved, i.e. 18.0%, as shown in Fig. 14c.

Lastly, Fig. 14d illustrates the overall energy conserved by HEED cluster heads when HEED is run alone, when HEED is run with CASE Compact and when HEED is run with CASE Compact, with the inclusion of rule-based cluster head selection. In rule-based cluster head selection, the reclustering frequency of CASE Compact is set as a function of the total cluster size for the cluster group, $Reclustering_{\text{prob}} \leftarrow \frac{clusterSize - numberOfConsequents}{clusterSize}$, to adapt the reclustering frequency to the level required by HiCoRE. Four simulation runs are conducted for each of the three cases with 50 nodes and a confidence value of 0.7 for a duration of 25 min in each run (seed values 124,755, 20,000, 25,000, 30,000 for runs 1, 2, 3 and 4, respectively). The results demonstrate that by using rule-based cluster head selection

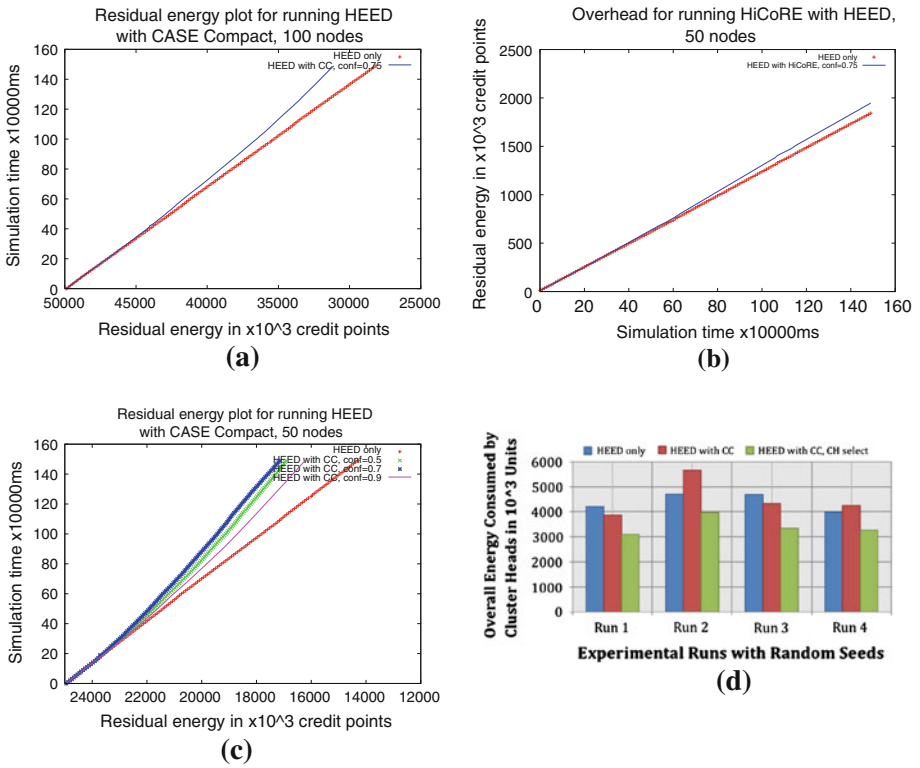


Fig. 14 Experimental results for running HEED with CASE Compact. **a** Running CASE Compact (HiCoRE) with 100 nodes. **b** Overhead of running HiCoRE. **c** Overall energy conserved for 50 nodes at different confidence threshold. **d** Overall cluster heads' energy consumption

with CASE Compact, the overall cluster head energy levels can be improved up to 42.5% (5,700−4,000/4,000) in run 2 when compared to running with CASE Compact without the selection. These savings are a consequence of the on-demand nature of reclustering that is now imposed for CASE Compact to further extend network lifetime.

7 Conclusion and future work

Utilising the limited energy of the battery-powered sensor nodes in wireless sensor networks is crucial to realise the great potential of using these networks in many significant applications. In this paper, we have presented our CASE and CASE compact frameworks to conserve energy in wireless sensor networks. Implementation details, which will be of great interest to both practitioners and researchers, are provided. The experimental results have provided an evidence of the potential for energy savings that can be achieved through CASE and CASE Compact.

Future directions include deployment in dense wireless sensor networks with a magnitude of thousands of nodes and a thorough experimental study in the field for testing the scalability of our techniques. Furthermore, we plan to use query-driven approach for further improving the performance of the proposed techniques.

8 Appendix 1: Code listing

```
t = new Timer();
t.schedule(new TimerJob(context_tag),timeToTrigger);
```

Listing 1: Schedule context_tag

```
int addr = (short) Integer.parseInt(argv[argv.length - 1], 16);
RecvMsg packet = new RecvMsg();
packet.set_source(0);
packet.set_action(TRANSMIT_RATE);
packet.set_args_ss_args_interval(1000);
MoteIF mote = new MoteIF(PrintStreamMessenger.err);
mote.send(addr, packet);
```

Listing 2: Trigger message

```
task void cmdInterpret() {
        struct RecvMsg * cmd =
       (struct RecvMsg *) msgRecv->data;
        cmd->source = TOS_LOCAL_ADDRESS;
        switch (cmd->action) {
            case LED_ON:
                call Leds.greenOn();
                break;
            case LED_OFF:
                call Leds.greenOff();
                break;
             case TRANSMIT_RATE:
                call ProcessCmd.changeTransmitRate
         (cmd->args.ss_args.interval);
                break;
             case PACKET_TYPE:
                atomic {
                    pType = cmd->args.ss_args.interval;
                    packetReadingNumber = 0;
                }
                break;
            case SLEEP:
                timeLeft = cmd->args.ss_args.interval;
                call ProcessCmd.sleepMote(timeLeft);
                break;
        }
        pending = 0;
        signal ProcessCmd.done(msgRecv, SUCCESS);
    }
```

Listing 3: Command interpret module

```
1  event result_t InterceptSurgeMsg.intercept
2                 (msg, payload, payloadLen) {
3    ...
4    if (TOS_LOCAL_ADDRESS != 0)
5      call Mining.putDataToBatch
6                 (msg->addr, msg->reading);
7    ...
8  }
```

Listing 4: Batch transaction

```
1
2  // light packet sent at timer fire
3  task void SendData() {
4
5   SurgeMsg *pReading;
6   // construct data packet
7   ...
8   if ((call Send.send(&gMsgBuffer,
9       sizeof(SurgeMsg))) != SUCCESS)
10          atomic gfSendBusy = FALSE;
11 }
12
13 // signal radio busy status, packet not sent.
14 task void triggerTask() {
15  atomic gfSendBusy = TRUE;
16 }
```

Listing 5: Packet send procedure

9 Appendix 2: CASE modules

9.1 CASE-PC modules

The CASE-PC modules are Java classes that run on the base station. The implementation has been captured in the form of a class diagram. The class diagram in Fig. 15 shows the classes used to implement main modules in Fig. 1.

To enable users of CASE to program new context_tags and monitor the triggers that have been activated, a GUI front-end has been implemented as illustrated in Fig. 16. Upon start-up, the front-end is initialised with sensory values shown in Fig. 16a. These sensory inputs are then analysed to yield contextual information. When a context_tag has been discovered, the triggering process begins as illustrated in Fig. 16b and the trigger results in commands sent to related sensors to conserve their energy, as shown in Fig. 16c. Lastly, Fig. 16d shows new commands sent upon getting new contextual data to return slept sensors to their default operations. This front-end is implemented using Java 1.1.

9.2 CASE TinyOS modules

In order for a mote to respond to trigger messages sent from the base station node, the packet types and modules to handle received packets are defined as follows:

– **Message type** In our implementation, a sensor node is programmed to respond to the following types of trigger commands: (1) sleep; (2) reduce transmission load; (3) reduce transmission frequency; and (4) default sensing. To handle these commands, the sensor

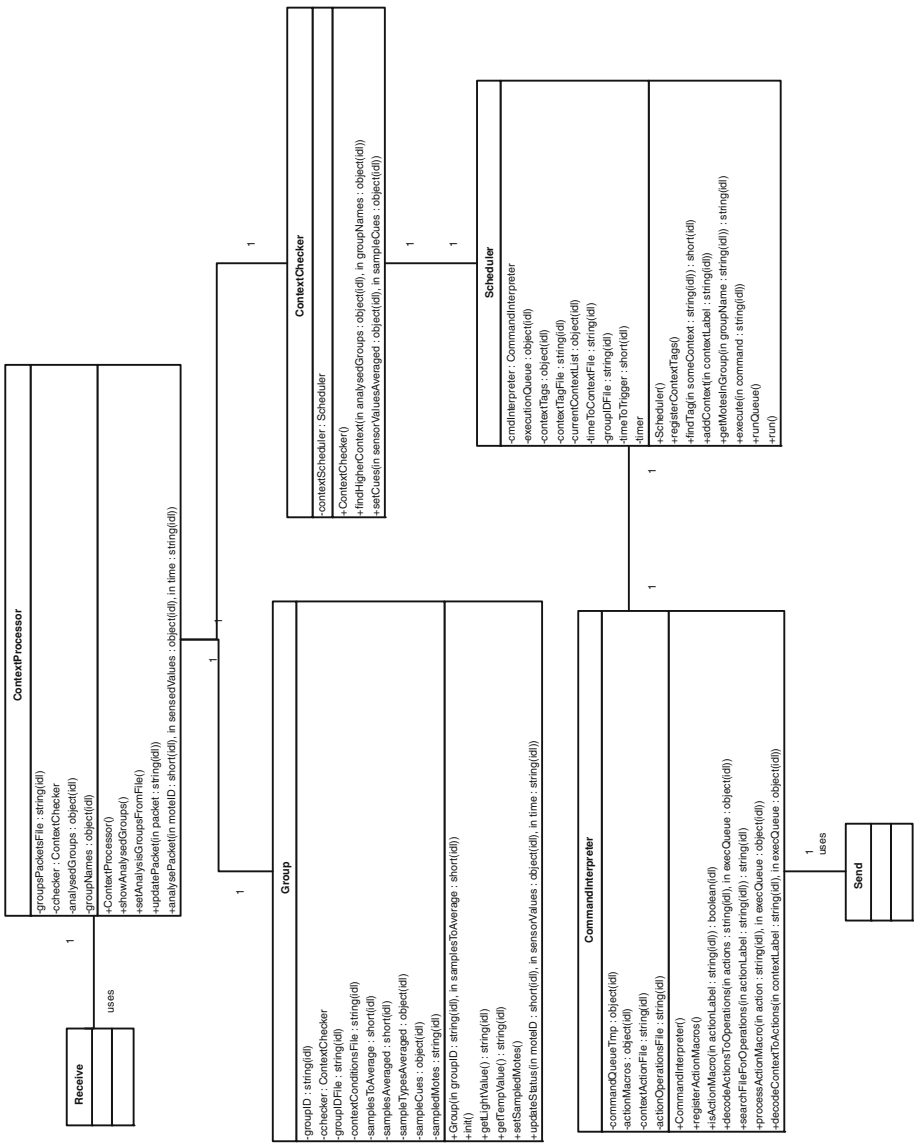


Fig. 15 Class diagram

node is programmed to receive a generic packet trigger message from the base station. This message type consists of:

- **Source id** This refers to the address of the base station node that has sent the packet in integer form.
- **Action** This refers to the trigger command that needs to be executed. This can be represented by an integer datatype, such as 5 for SLEEP.
- **Parameters** This refers to the parameters that follow the trigger command. For instance, the TRANSMIT_RATE command would require the actual rate to transmit in milliseconds, e.g. 1,000 ms.

Fig. 16 CASE front-end **a** Initialisation. **b** Trigger in process. **c** Context_tag 'Nighttime' triggered. **d** Context_tag 'Daytime' triggered

– **Trigger handler** The trigger command handler within the CASE-mote implementation uses the **action** parameter in the trigger message to determine the type of operation that is to be performed next on the mote. This can be implemented using a switch statement

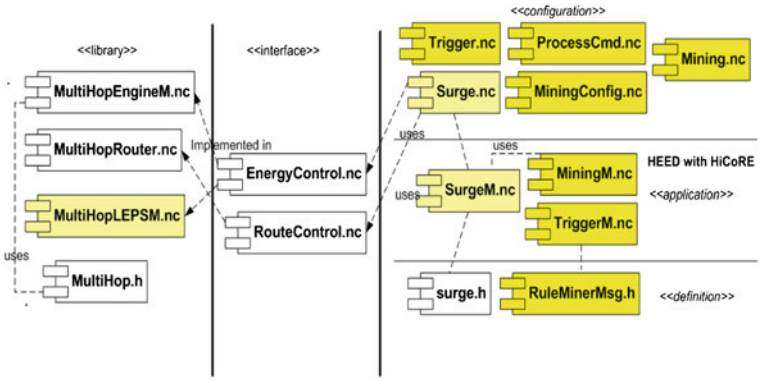


Fig. 17 HEED with CASE Compact code structure

(code fragment as listing 3 in appendix 1) to affect the current sensing and transmitting operations that the sensor currently runs.

9.3 HiCoRE implementation with HEED

The implementation of CASE Compact is performed on top of the HEED implementation. The HiCoRE algorithm is installed on all sensor nodes but is only executed on the cluster heads when they have been nominated through HEED. In implementation terms, it occupies an additional 9,446 bytes in nesC code when pre-programmed into all nodes that also run HEED with Surge. The code structure for this implementation is shown in Fig. 17. The modules pertaining to the implementation of HiCoRE have been highlighted by yellow (to represent new modules installed) and light yellow (to represent existing module of HEED with Surge that have been modified) in Fig. 17.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: 20th International conference on very large data bases. Chile, pp 487–499
2. Apiletti D, Baralis E, Cerquitelli T (2010) Energy-saving models for wireless sensor networks. Knowl Inf Syst. doi:10.1007/s10115-010-0328-6
3. Arici T, Gedik B, Altunbasak Y, Liu L (2003) Pinco: a pipelined in-network compression scheme for data collection in wireless sensor networks. In: 12th International conference on computer communications and networks. Texas, pp 539–544
4. Baker CR, Armijo K, Belka S, Benhabib M, Bhargava V, Burkhart N, Minassians AD, Dervisoglu G, Gutnik L, Haick MB, Ho C, Koplow M, Mangold J, Robinson S, Rosa M, Schwartz M, Sims C, Stoffregen H, Waterbury A, Leland ES, Pering T, Wright PK (2007). Wireless sensor networks for home health care. In: 21st Advanced information networking and applications workshops. Niagara Falls, pp 832–837
5. UC Berkeley (2003) Serial-line communication in tinyos-1.1. http://www.tinyos.net/tinyos-1.x/doc/serialcomm/description.html (Accessed 3/9/2008)
6. Burrell J, Brooke T, Beckwith R (2004) Vineyard computing: sensor networks in agricultural production. IEEE Perv Comput 3(1):38–45
7. Chen JY, Pandurangan G, Xu D (2005) Robust computation of aggregates in wireless sensor networks: distributed randomized algorithms and analysis. In: 4th International symposium on information processing in sensor networks. California, pp 348–355

8. Chong SK, Gaber MM, Krishnaswamy S, Loke SW (2008) A rule learning approach to energy efficient clustering in wireless sensor networks. In: 2nd International conference on sensor technologies and applications. Cap Esterel, pp 329–334
9. Chong SK, Krishnaswamy S, Loke SW (2005) A context-aware approach to conserving energy in wireless sensor networks. In: 3rd IEEE international conference on pervasive computing and communications workshops. Hawaii, pp 401–405
10. Chu D, Deshpande A, Hellerstein JM, Hong W (2006) Approximate data collection in sensor networks using probabilistic models. In: 22nd International conference on data engineering. Atlanta, pp 48
11. Dalton AB, Ellis CS (2003) Sensing user intention and context for energy management. In: 9th Workshop on hot topics in operating Systems. USENIX, Hawaii, pp 151–156
12. Deshpande A, Guestrin C, Madden SR, Hellerstein JM, Hong W (2004) Model-driven data acquisition in sensor networks. In: 30th Very large data base conference. Toronto, pp 588–599
13. Elnahrawy E, Nath B (2004). Context-aware sensors. Lecture notes in computer science. Springer, 2920, pp 27–93
14. Gedik B, Liu L, Yu PS (2007) Asap: an adaptive sampling approach to data collection in sensor networks. IEEE Trans Parallel Distrib Syst 18(12):1766–1782
15. He T, Krishnamurthy S, Stankovic JA, Abdelzaher T, Luo L, Stoleru R, Yan T, Gu L, Hui J, Krogh B (2004) Energy-efficient surveillance system using wireless sensor networks. In: 2nd International conference on mobile systems, applications and services. Boston, pp 270–283
16. Hefeeda M, Bagheri M (2007) Wireless sensor networks for early detection of forest fires. In: IEEE International conference on mobile adhoc and sensor systems. Pisa, pp 1–6
17. Heinzelman WR, Chandrakasan A, Balakrishnan H (2000) Energy-efficient communication protocol for wireless microsensor networks. In: 33rd Annual Hawaii international conference on system sciences. Maui, pp 2–12
18. Hill J, Szewczyk R, Woo A, Hollar S, Culler D, Pister K (2000) System architecture directions for networked sensors. In: 9th International conference on architectural support for programming languages and operating systems. New York, pp 93–104
19. Krishnamachari B, Estrin D, Wicker S (2002) The impact of data aggregation in wireless sensor networks. In: 22nd International conference on distributed computing systems workshops. Vienna, pp 575–578
20. Liu H, Lin Y, Han J (2011) Methods for mining frequent items in data streams: an overview. Knowl Inf Syst 26:1–30
21. Loo KK, Tong I, Kao B, Cheung D (2005) Online algorithms for mining inter-stream associations from large sensor networks. In: Pacific-Asia conference on knowledge discovery and data mining. Hanoi, pp 143–149
22. Madden S, Franklin MJ (2002) Fjording the stream: an architecture for queries over streaming sensor data. In: 18th International conference on data engineering. San Jose, pp 555–566
23. Manjhi A, Nath S, Gibbons PB (2005) Tributaries and deltas: efficient and robust aggregation in sensor network stream. In: Special interest group on management of data. Baltimore, pp 287–298
24. McCauley I, Matthews B, Nugent L, Mather A, Simons J (2005) Wired pigs: Ad-hoc wireless sensor networks in studies of animal welfare. In: 2nd IEEE workshop on embedded networked sensors. Sydney, pp 29–36
25. Nath S, Gibbons PB, Seshan S, Anderson ZR (2004) Synopsis diffusion for robust aggregation in sensor networks. In: ACM conference on embedded networked sensor systems. Baltimore, pp 250–262
26. Passos RM, Nacif JA, Mini RAF, Loureiro AAF, Fernandes AO, Coelho CN (2006) System-level dynamic power management techniques for communication intensive devices. In: International conference on very large scale integration. Nice, pp 373–378
27. Perillo M, Ignjatovic Z, Heinzelman W (2004) An energy conservation method for wireless sensor networks employing a blue noise spatial sampling. In: International symposium on information processing in sensor networks. California, pp 116–123
28. Petrovic D, Shah RC, Ramchandran K, Rabaey J (2003) Data funneling: routing with aggregation and compression for wireless sensor networks. In: 1st IEEE international workshop on sensor network protocols and applications. California, pp 156–162
29. Qin B, Xia Y, Prabhakar S (2010) Rule induction for uncertain data. Knowl Inf Syst. doi:10.1007/s10115-010-0335-7
30. Shah RC, Roy S, Jain S, Brunette W (2003) Data mules: modeling a three-tier architecture for sparse sensor networks. In: 1st IEEE international workshop on sensor network protocols and applications. Seattle, pp 30–41
31. Shnayder V, Hempstead H, Chen B, Allen GW, Welsh M (2004) Simulating the power consumption of large-scale sensor network applications. In: 2nd International conference on embedded networked sensor systems. Baltimore, pp 188–200

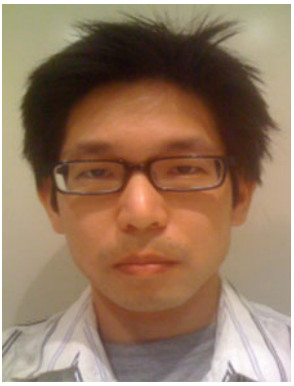
32. Tavakoli A, Zhang J, San SH (2005) Group-based event detection in undersea sensor networks. In: 2nd International workshop on networked sensing systems. San Diego. http://www.cs.berkeley.edu/arsalan/Papers/GroupDetection_INSS.pdf (Accessed 10/11/08)
33. Tian D, Georganas ND (2002) A node scheduling scheme for large wireless sensor networks. Wirel Commun Mobile Comput J 3(2):271–290
34. Tulone D, Madden S (2006) Paq: time series forecasting for approximate query answering in sensor networks. In: 3rd European conference on wireless sensor networks. Zurich, pp 21–37
35. Vasilescu I, Kotay K, Rus D, Dunbabin M, Corke P (2005) Data collection, storage, and retrieval with an underwater sensor network. In: 3rd International conference on embedded networked sensor systems. California, pp 154–165
36. Widom J, Ceri S (1996) Active Database Systems: triggers and rules for advanced database processing. Morgan Kaufmann. ISBN: 1558603042
37. Ye F, Zhong G, Cheng J, Lu S, Zhang L (2003) Peas: a robust energy conserving protocol for long-lived sensor networks. In: 23rd International conference on distributed computing systems. Providence, pp 28–37
38. Younis O (2005) iheed source code. http://www.cs.purdue.edu/homes/fahmy/software/iheed/tinyos-1.x/ (Accessed 03/11/2007), 2005
39. Younis O, Fahmy S (2004) Heed: a hybrid, energy-efficient, distributed clustering approach for ad-hoc sensor networks. IEEE Trans Mobile Comput 3(4):366–379
40. Zhao J, Govindan R, Estrin D (2003) Computing aggregates for monitoring wireless sensor networks. In: 1st IEEE international workshop on sensor network protocols and applications. California, pp 139–148

Author Biographies

Suan Khai Chong is currently a software developer at Service Network Systems Branch in Centrelink IT. He received his PhD from Monash University, Caulfield Campus in 2009. During his PhD, he has held internship positions at the University of Queensland, CSIRO and the Department of Primary Industries. Khai's research interests include wireless sensor networks, context-aware computing and data-stream mining.

Mohamed Medhat Gaber has research interests in data stream mining, wireless sensor networks, context-aware computing and health informatics. He received his PhD from Monash University, Australia in 2006. He then held appointments with the University of Sydney, CSIRO, and Monash University. Mohamed is currently a senior lecturer in the School of Computing, University of Portsmouth, UK. He has published more than 60 papers in the areas of data mining and machine learning. Mohamed is the co-editor with J. Gama of the book: Learning from Data Streams: Processing Techniques in Sensor Networks, published by Springer in 2007, the book: Knowledge Discovery from Sensor Data published by CRC in 2008, and the editor of the book Scientific Data Mining and Knowledge Discovery: Principles and Foundations published by Springer in 2009. Mohamed has served in the program committees of numerous international conferences and workshops in the area of data mining and context-aware computing.

Shonali Krishnaswamy is an Associate Professor in the Faculty of Information Technology at Monash University, Australia. Shonali holds a PhD (Computer Science, 2003) and Master of Computing (1998) from Monash University, a Bachelor Science (Computer Science, 1996) from the University of Madras, India, and a Graduate Certificate in Higher Education (2006) from Monash University. Shonali has received the following awards/fellowships since commencing her academic career in 2003—Vice Chancellor's Award for Excellence in Research by an Early Career Researcher (2008), Faculty of Information Technology Early Career Researcher Award (2008), IBM Innovation Award (Unstructured Information Management Architecture (2008), and ARC Australian Post Doctoral (APD) Fellowship (awarded by the Australian Research Council) (2003–2008). Shonali's research is broadly the area of Distributed, Mobile and Pervasive Computing Systems where her focus on developing intelligent applications that aim to service real-time information needs while having to function in highly dynamic and resource-constrained environments. Her specific expertise is in Mobile and Ubiquitous Data Stream Mining, Service Oriented Computing and Mobile Software Agents.



Seng Wai Loke is a Reader and Associate Professor at the Department of Computer Science and Computer Engineering in La Trobe University. He received his PhD from the University of Melbourne, Australia. He leads the Pervasive Computing Group at La Trobe University, and has authored 'Context-Aware Pervasive Systems: Architectures for a New Breed of Applications' published by Auerbach (CRC Press), Dec 2006. He has (co-)authored more than 210 research publications, with work on context-aware computing, and mobile and pervasive computing.