Mobile Netw Appl (2011) 16:194–213
DOI 10.1007/s11036-010-0268-0

# Complex Event Detection in Extremely Resource-Constrained Wireless Sensor Networks

**Michael Zoumboulakis · George Roussos**

Published online: 9 October 2010


**Abstract** Complex Events are sequences of sensor measurements indicating interesting or unusual activity in the monitored process. Such events are ubiquitous in a wide range of Wireless Sensor Network (WSN) applications, yet there does not exist a common mechanism that addresses both the considerable constraints of WSNs and the specific properties of Complex Events. We argue that Complex Events cannot be described using standard threshold-based or composite logic approaches and attempting to represent them as such can lead to unpredictable execution cost while detection accuracy suffers from erroneous recording of observations which are common in WSNs. To address this, we develop a family of Complex Event Detection (CED) algorithms based on online symbolic conversion of sensor readings. With fixed execution cost and modest resource requirements, the CED algorithms cater for exact, approximate, non-parametric, multiple and probabilistic detection that is neither application nor data dependent. Overall, full implementation and simulations provide experimental evidence of the advantages of the proposed approach. We find that the proposed algorithms minimise configuration, promote unattended operation and complement the goal of prolonged lifetime—factors that satisfy the long-term research vision predicting Internet-scale WSNs comprising billions of devices.

**Keywords** wireless sensor networks · complex event detection · integer techniques

M. Zoumboulakis (✉)
Birkbeck College, University of London,
23-29 Emerald Street, London WC1N 3QS, UK
e-mail: mz@dcs.bbk.ac.uk

G. Roussos
Birkbeck College, University of London,
Malet Street, London WC1E 7HX, UK
e-mail: gr@dcs.bbk.ac.uk

## 1 Introduction

Complex Events are sequences of sensor measurements indicating interesting or unusual activity in the process monitored by the Wireless Sensor Network (WSN). Detecting such events with thresholds is problematic for several reasons including sensitivity to faulty readings, unnecessary complexity of performing similarity searches and inability of specifying unknown events. Furthermore, threshold-based detection has a variable execution cost that depends on the complexity of the event expression and cannot be determined at compile-time. To address these drawbacks we develop a data-mining inspired solution in the form of a family of Complex Event Detection (CED) algorithms based on online symbolic conversion of sensor readings. With fixed execution cost and modest resource requirements, the CED algorithms cater for exact, approximate, non-parametric, multiple and probabilistic detection that is neither application nor data dependent.

The target platform for the proposed solution is the lower end of the WSN spectrum that comprises inexpensive devices with severe resource constraints—RAM and program flash in the order of tens of Kilobytes (KB), embedded ultra low-power Microcontroller and power harvested from strong electromag-

netic emissions such as the WISP platform [3, 55]. Failures are frequent in this type of WSN and runtime behaviour is somewhat unreliable and usually compensated by high network density and local coordination. The CED algorithms operate efficiently within the resource envelope by requiring under 20 ms of active CPU time with less than 1KB RAM footprint. Moreover, the algorithms fit well with existing WSN protocols without incurring a communication overhead in comparison with threshold techniques or requiring modifications in the standard programming interface for reactive applications.

We begin the discussion in Section 1.1 with a detailed treatment of the problem and an outline of the contributions. The ubiquity of Complex Events in WSNs is illustrated by reviewing related work and alternative solutions in Section 1.2 while Section 2 delves deeper into the characteristics of Complex Events. Section 3 presents the family of algorithms for CED and Section 5 discusses experimental evaluation over four real-world sensor data sets. Finally, Section 4 describes the techniques that make the implementation efficient and suitable for severely resource-constrained WSN nodes.

### 1.1 Problem statement & contributions

Typical WSN applications users frequently wish to be informed of interesting changes but may be able to describe what constitutes interesting using only general constructs or a high-level language. Converting an event description from such a high-level language to a set of programming instructions can be challenging and costly. The option of using composite event calculus for the lower level representation, implies a lack of a compile-time guarantee regarding the processing cost of event interests submitted at run-time. Furthermore, this approach can exclude events that are similar to the one specified while it may include events that spuriously match the specification, for example, when faulty readings cause a false positive. In other situations a user cannot provide specific information about the event, but can only describe it as a pattern that deviates from normalcy. This is problematic since it usually requires rich state for detection which can be costly in terms of memory and power resources.

Moreover, event detection should be performed in-the-network since local computation can be orders of magnitude cheaper than radio communication [30]. Therefore the central problem tackled in this article is the provision of algorithms for efficient, in-network Complex Event Detection (CED). Before we discuss further the properties of Complex Events and the proposed solution, let us examine three such event examples shown in Fig. 1.

Figure 1a shows a seismic event from a real-world deployment in an active volcano in Ecuador [50]. This event comprises approximately 1,100 data points. Figure 1b shows a chemical dispersion event that spans the 2-dimensional space and comprises approximately 10,000 points. It is constructed by the model proposed by Ishida et al. [22] and is similar to target events considered by applications such as Gunatilaka et al. [16] and Chin et al. [9]. Finally, Fig. 1c shows a *gesture* event from robot data sensed by a 3-axis accelerometer [24] and comprises approximately 1,000 data points. Gesture detection and classification [46] can be used to determine the context or activity of a user.

These three examples share a common characteristic: they reveal interesting activity that would be cumbersome to describe and inefficient to capture using traditional techniques such as thresholds and composite event calculus. The alternative calls for distance-based detection that is efficient, scalable and fault-tolerant. We therefore consider the following problem:

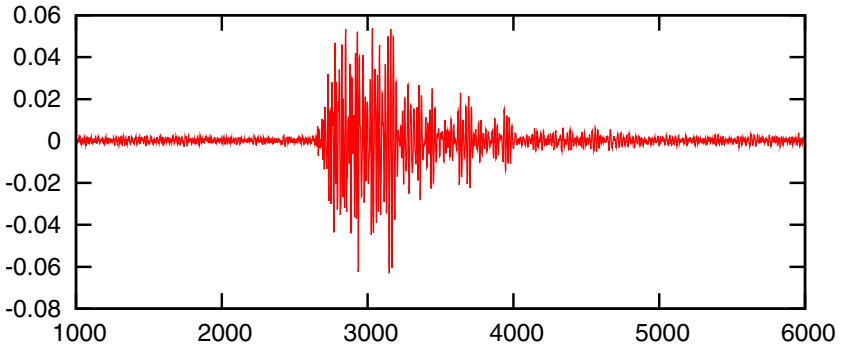
*Problem statement* The provision of a family of algorithms that address efficient Complex Event Detection (CED) in extremely resource-constrained WSNs. The investigation of the aforementioned problem from a performance, scalability and fault-tolerance perspective.

The motivating factor for this work is that Complex Events are ubiquitous across a number of WSN applications—a selection of which is presented in Section 1.2—yet there does not exist a mechanism that addresses both the specific constraints of WSNs and the peculiar properties of Complex Events. To this end, we make the following contributions:

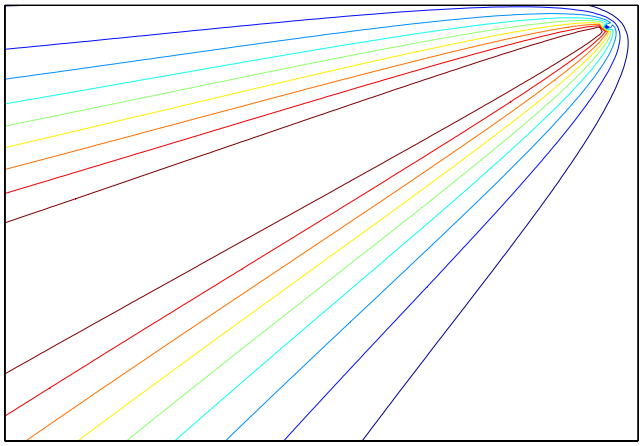
(i.) *Determine that traditional threshold-based and composite event approaches are not a good fit for the resource-constrained WSN environment.* Factors such as outliers, missing sensor values and performance considerations make detection using composite event techniques problematic (explored further in Section 2).

(ii.) *Propose a family of algorithms for Complex Event Detection (CED) in Wireless Sensor Networks* (WSNs) that is suitable for extremely resource-constrained nodes. The suitability is based on the fact that they can be used to successfully detect Complex Events across a number of applications, evaluated in Section 5, in an efficient manner with a fixed execution cost known at compile-time. The algorithms, fully

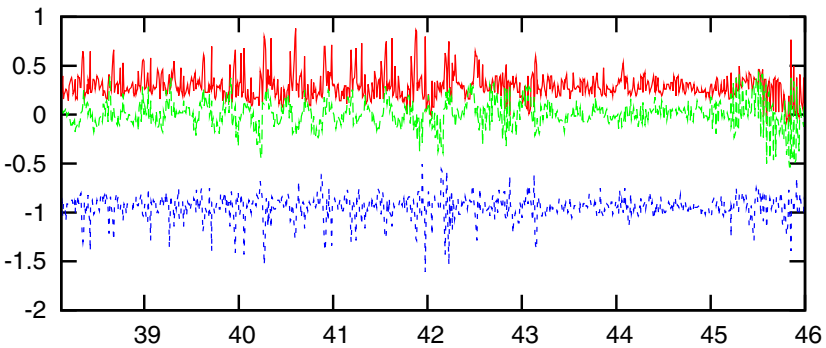
**Fig. 1** Complex event examples from different applications




(a) Example of a seismic event



(b) Example of a chemical dispersion event




(c) Example of a gesture event

implemented in TinyOS and made available to the WSN community, include:

(a.) *Exact or Approximate CED*: where the pattern is known beforehand to the user of the system.

(b.) *Non-parametric CED*: where the pattern does not need to be specified in advance. Instead, sensor nodes *learn* from a training portion of the data known to be normal.

(c.) *Multiple CED*: this makes detection scalable as event subscriptions increase by using an efficient data structure to store subscriptions.

(d.) *Probabilistic CED*: this can enhance detection by adding probabilities to both observed and unobserved Complex Events.

(iii.) *Propose a Dynamic Sampling Frequency Management (DSFM) algorithm*. Similar to Non-parametric CED, this algorithm employs a training phase to learn the sensed process dynamics, and use it to make autonomous local decisions to dynamically increase or decrease the

sampling frequency within application-defined bounds. Dynamically adjusting the frequency enables WSN nodes to conserve energy in periods of relative inactivity for times when interesting changes occur.

(iv.) *Introduce Integer Techniques* for efficient WSN programming (Section 4). We adapt [56] an established timing model that can guide development and highlight tradeoffs. In addition, we propose a collection of integer optimisations and illustrate their applicability to WSNs by presenting their impact to the efficiency of the CED algorithm.

(v.) *Integrate CED with a Publish/Subscribe Interface*. As well as providing a familiar interface for users, we show this can be done efficiently without requiring modifications to the underlying communication protocols.

Each of the above contributions is addressing a specific problem: for instance approximate matching enables users to search for events that are similar to other events. This cannot be easily achieved using the composite event alternative. Non-parametric detection enables users to search for interesting changes without supplying any event specification or other prior information. The efficiency of the Suffix Array structure makes CED scalable, by allowing nodes to store and search multiple patterns simultaneously. Probabilistic detection makes the system usable by applications with a need to attach probabilities to patterns. Integer techniques make the implementation computationally efficient. Finally, the CED functionality benefits from features of Publish/Subscribe without adding overhead to communication or requiring modification to protocols.

### 1.2 Related work

The main limitation of the majority of the research efforts described below is that they are tightly coupled to the specific application scenario considered making them data dependent. In contrast, our approach requires minimal or no configuration and has been shown to work efficiently and accurately in a number of real-world data sets.

Complex Events are evident in the Ecuador deployment that monitored the active volcano of Reventador [50], collecting data with microphones and seismic sensors to detect earthquakes, eruptions and seismo-acoustic events. The authors describe *long-period events* such as tremors or earthquakes that are present in the publicly available data. The application employs a custom event detection algorithm, based on two exponentially-weighted moving averages (EWMA) over the input signal with different gain settings. When the ratio between the two EWMAs exceeds a threshold, the node transmits an event report to the base station. If the base station receives events from 30% of the active nodes within a 10sec window, it considers the event to be well-correlated and initiates data collection. A limitation of the EWMA approach [51] is that often small tremors were detected but subsequent large earthquakes were missed. This is due to high sampling frequency that restricted nodes from sensing and sending simultaneously: on events nodes stopped sensing and started pushing data to the base station.

The approach described in Basha et al. [2] presents a model-based predictive system that aims to detect and predict river flood events in developing countries by deploying sensor networks around the basin area of rivers. The simplest model is based on statistical methods such as linear regression using a portion of data known to be normal. The project aims to cover vast geographical regions of approximately 10,000 $km^2$ and predict a Complex Event of interest using a distributed model driven by the collected data. The main drawback of this approach is that it assumes a tiered architecture where resource-constrained sensor nodes transmit summaries and statistics of raw data to a set of computation nodes. The latter determine the correctness of the data, feeds it to the model for prediction and may request additional data from sensors to reduce uncertainty. A somewhat similar tiered system is PRESTO [10] which employs ARIMA (Auto Regressive Integrated Moving Average) time series forecasting models and performs anomaly detection by comparing predicted values to sensor observations.

The work in Xue et al. [52] agrees with our assessment regarding the unsuitability of thresholds for WSN event detection and, similar to our approach, converts detection to a pattern matching problem. The authors suggest the use of contour maps to display the distribution of attribute values in the network and employ contour map matching to determine whether a user-supplied pattern matches the one produced by the nodes. The application scenario is event detection in coal mines, monitoring for the occurrence of gas, dust and water leakage as well as high/low oxygen density regions. A limitation of this approach is that it assumes users capable of perfectly describing the Complex Event of interest as distributions of an attribute over space and variations of this distribution over time incurred by the event. As we will see in Section 3, we offer Non-parametric CED that caters for unknown

thresholds and does not burden the user with providing Complex Event specifications.

The approach described in Huang et al. [20], proposes a Principal Component Analysis (PCA) method for detecting anomalies with complex thresholds in a distributed manner. Nodes send their readings to a coordinator node that is responsible for firing a trigger based on the aggregate behaviour of a subset of nodes. The individual nodes perform filtering such that they send readings only when measurements deviate significantly from the last transmitted data. With respect to detection, they propose two window triggers that capture anomalous behaviour over fixed and varying time series windows. Although the approach is aimed at detecting unusual network traffic patterns, it could apply to certain WSN applications. The main criticism is that it creates single points of failure by assigning the coordinator role to nodes.

Another application area where Complex Events are observed is in wearable computing and Body Sensor Networks (BSN). The work in Stiefmeier et al. [46] targets the recognition of users' context by online detection and classification of *gestures*: the movement of limbs in space. Accelerometer data is converted to strings and, similar to our approach, detection is performed using string matching. However, contrary to the family of CED algorithms we propose, the detection is performed by a desktop-class computer using techniques such as Hidden Markov Models and k-Nearest Neighbour. A related system is described in Katsiri et al. [23] where ECG data collected by low-end sensor nodes is used for real-time heart rate variability analysis.

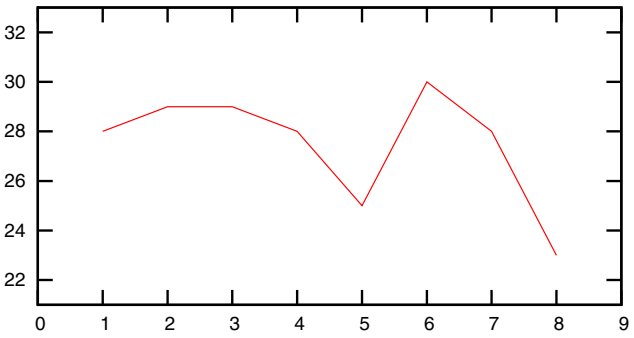
Other application areas with Complex Event are: Soil Moisture Monitoring [5, 37, 48], Precision Agriculture ([4, 12], Structural Health Monitoring [7, 27, 29, 33, 40, 47], Oceanographic Monitoring [18]), Pervasive Healthcare [15, 28, 34] and Disaster Detection and Response [1, 6, 11, 35, 45]. Complex events in such applications are usually detected by custom-made approaches that provide unique functionality for their requirements and no common representation or approach exists to address the problem of data and application independent Complex Event Detection (CED). This forms the motivation for our work, described in the following sections.

## 2 Characteristics of complex events

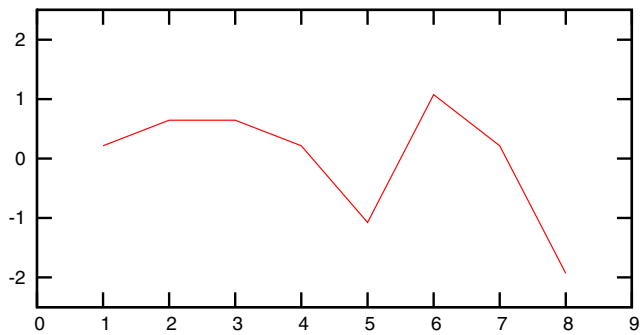
Composite event calculus, as defined in Chakravarthy et al. [8], is fit for numerous distributed database systems but it severely limits the detection capabilities for real-world WSN applications. In this context, a composite event is constructed from other primitive events. In contrast, a Complex Event is an interesting or unusual pattern in the data sensed and processed by WSN nodes. This pattern describes a real-world state and usually it cannot be decomposed to constituent primitive events.

The use of composite event calculus tends to lead to expressions that grow in proportion to the length of the sequence *and* the complexity of the event interest. This problem has been identified by others arguing for composite event approaches in WSNs: for instance [44] states that "*some Boolean expressions may convert to an exponentially long Conjunctive Normal Form* [...] *a longer expression is undesirable because it requires more computation*". We concur with this evaluation and argue that although such expressions are linear in nature, they scale poorly as the number and complexity of user interests grow. Furthermore, they suffer from the following weaknesses:

– Sensitivity to outliers and missing values which are very common in sensor data sets due to inexpensive sensors,




(a) Temperature pattern on a Celsius scale




(b) Identical pattern z-standardised

**Fig. 2** Illustration of scale differences among two identical patterns

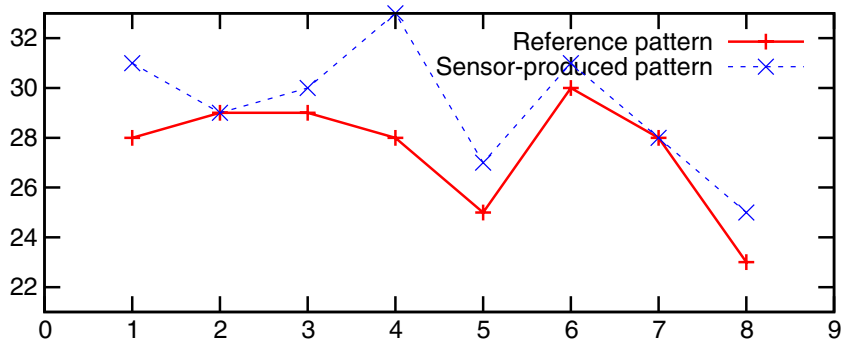



**Fig. 3** An illustration of two time series sequences: a reference pattern that represents an event interest by a user and a sensor produced sequence, such as temperature readings

- Difficulty of performing similarity searches in a straightforward manner. We will illustrate this shortly with a specific example,
- Inability of coping with unknown thresholds, for instance the case where an event is not describable. This is a common issue in WSNs especially in new deployments: it is sometimes difficult for users to describe numerically or programmatically a pattern of interest.
- Inability of handling differences in scale in an efficient manner. The example shown in Fig. 2 illustrates this: the two patterns are identical but one of them is z-standardised. The use of thresholds necessitates two event expressions with different threshold values.

In addition, composite event approaches suffer from an inherent complexity of dealing with similarity searches that translate to long expressions of disjunctive or conjunctive form. Consider the two time series shown in Fig. 3, where the solid line represents an interest by a user and the dotted line represents a sequence of sensor readings. For this example we assume that the latter *approximately* matches the reference pattern. We further assume that a user is interested in a similarity match and describes it as a composite event expression where each data point in the sensor sequence is at most three units away from the corresponding data point in the reference pattern.

But such a composite expression does not cater for the case where the fourth data point differs by five units while the rest are all equal. Or the case where the last data point is missing altogether from the sequence. This reveals a need to define similarity as a form of *distance* of each data point in the interest from the corresponding point in the sensor-produced sequence. Since we are dealing with streaming time series data we argue that describing event matching as a similarity problem is more suitable than arbitrary long expressions combining clauses by disjunction or conjunction. Therefore by enforcing a similarity measure to event matching we effectively bound the cost of evaluating event occurrences.

Composite event specifications can expand to expressions of arbitrary length and complexity, but the similarity measure of our approach has a fixed execution cost, linear with respect to the size of the time series. Conversely, a composite event expression depends not only on the length of the time series but also on the complexity of the event interest. We therefore conclude that our approach does not suffer from the code complexity and state persistence necessary in composite event calculus. As we will see in the sections to follow our approach is efficient, accurate and scalable without penalising the user with the task of designing arduous composite event specifications.

## 3 A family of algorithms for CED

The basis for the family of CED algorithms is online *Symbolic Conversion*: the conversion of streaming sensor measurements to strings of characters. Event detection becomes an instance of the pattern matching problem using a distance metric that signifies a departure from Boolean semantics and allows for similarity searches between user-supplied and sensor-produced patterns. Furthermore, the distance metric facilitates the identification of interesting or unusual Complex Events without requiring any prior information or threshold values from the user.

Equating the detection problem to pattern matching simplifies the development of reactive WSN applications by minimising the amount of data analysis effort required at pre-deployment. For instance, by offering a selection of CED algorithms users can decide which

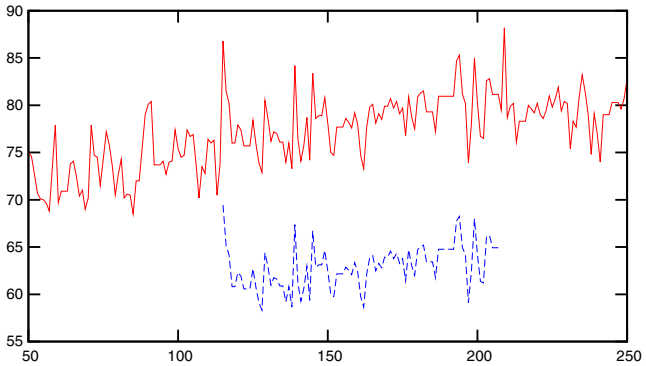
algorithm to employ at runtime lifting the requirement for pre-deployment data acquisition. The choice of CED algorithm depends on the usage scenario rather than the underlying data characteristics: a user searching for a previously observed event can supply it as a template for the exact or approximate CED algorithm while another user can choose to let nodes determine which Complex Events are sufficiently unusual, given a training window. It is possible to mix CED algorithms on the same WSN or even run different CED algorithms in parallel at the node level. This provides a degree of data and application independence and preliminary confirmation is provided by the evaluation results of Section 5 that applied the CED algorithms to data sets of varying characteristics. In addition, working with string representation of sensor measurements enables the use of a large collection of string algorithms [17] primarily inherited from the bioinformatics research community. This means the core functionality of CED is easily extensible, if deemed necessary by applications with specific requirements. For example a user can select an alternative string distance function if that provides better detection performance for the nature of sensor data.

With respect to the conversion of the sensor measurements to strings, we employ the established Symbolic Aggregate Approximation (SAX) [32] data mining algorithm. We treat SAX as a black box to which we pass a numeric sensor sequence and it returns a reduced string representation. We then employ the SAX string distance metric to determine equality or similarity between patterns—henceforth the term *pattern* will be used interchangeably with Complex Event. Due to space limitations, we will not review SAX in more detail here and refer interested readers to the available literature on SAX [25, 26, 32] and our WSN-specific implementation [53].

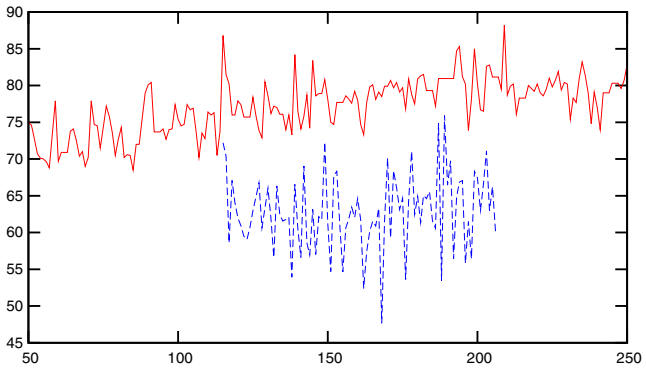
The suite of Complex Event Detection (CED) algorithms to follow, is fully integrated with a flexible content-based Publish/Subscribe (Pub/Sub) system for TinyOS [19]. This offers a familiar interface to users of a reactive system while employing standard fault-tolerant TinyOS protocols such as Trickle [31]. Pub/Sub is an attractive communication mechanism since it caters for asynchronous and anonymous communication accommodating disconnected nodes and the address-free nature common in a number of WSN applications. The integration of CED with Pub/Sub shows that our approach to event detection does not require changes to the communication mechanism of a reactive WSN, in terms of programming and use of a standard interface. Furthermore, there is no additional communication overhead in comparison with a composite event calculus threshold-based technique.

### 3.1 Exact and approximate CED

Exact and approximate CED represent the case where a user knows and is able to describe the pattern of interest. Application such as the one described in Stiefmeier et al. [46] are prime candidates for this functionality. Furthermore, any application where detection depends on classifying a sensor signal segment to a predetermined category can benefit from Exact and Approximate CED. The main difference between the two algorithms is that with approximate CED users can perform similarity searches in a straightforward manner. Consider the example of Fig. 4a that shows an exact match. Note that although the two patterns are of different magnitude they still match exactly, for instance the string distance between them is zero. The second example (Fig. 4b) shows an approximate match




(a) Exact CED




(b) Approximate CED

**Fig. 4** Examples of exact and approximate CED—the *dotted line* represents the pattern

with the two patterns having a non-zero distance that satisfies a threshold.

**Algorithm 1** Exact Complex Event Detection (ECED) Algorithm

**Require:** `user-pattern` $\neq \varepsilon$
1: **if** `user-pattern` is numeric **then**
2: $\bar{r} \leftarrow$ call SAX(`user-pattern`)
3: **else**
4: $\bar{r} \leftarrow$ `user-pattern`
5: **end if**
6: **repeat**
7: $\bar{u} \leftarrow$ call SAX(`sensor-values[]`)
8: $\delta \leftarrow \|\bar{u} - \bar{r}\|_S$
9: **until** $\delta == 0$
10: call Notify and **goto** line 6

The procedural steps for exact and approximate CED are listed in Algorithms 1 and 2 respectively. The function call to SAX (line 6) performs the symbolic conversion according to the Symbolic Aggregate Approximation (SAX) [32] algorithm. The $\|\bar{u} - \bar{r}\|_S$ notation (line 8) represents the string distance between $\bar{u}$ and $\bar{r}$, which stand for the string representation of the sensor-produced values and the string representation of the reference pattern supplied by the user respectively. This string distance metric lower-bounds the Euclidean distance and is calculated by a lookup on a static *breakpoints* table. The lower-bounding property refers to a guarantee that the string distance will always be less (or equal) than the Euclidean distance between the sensor measurements and the user-supplied pattern. The importance of lower bounding is that it guarantees no false dismissals; the proof is beyond the scope of this article, but the interested reader can refer to Faloutsos et al. [14]. The formula employed to obtain the string distance is formally given by Keogh et al. [25]:

$$\|\bar{u} - \bar{r}\|_S = \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} (dist(\bar{u}_i, \bar{r}_i))^2} \quad (1)$$

The $dist()$ function refers to a lookup to the breakpoints table $B$ (a sample is presented in Table 1), $\bar{u}, \bar{r}$ denote the string representations of sensor measurement segments $u$ and $r$ respectively, and $n$ and $w$ represent the length of data points in $u, r$ and $\bar{u}, \bar{r}$ respectively. These breakpoints divide the area under a Gaussian curve into a number of equiprobable regions and the table is the same one used to obtain the string (stored in local memory of each node). Although other distance metrics can be used, we have found through empirical evaluation the SAX distance metric suits well the task of CED.

**Algorithm 2** Approximate Complex Event Detection (ACED) Algorithm

**Require:** `user-pattern` $\neq \varepsilon$
**Require:** *theta* $\neq \varepsilon$
1: **if** `user-pattern` is numeric **then**
2: $\bar{r} \leftarrow$ call SAX(`user-pattern`)
3: **else**
4: $\bar{r} \leftarrow$ `user-pattern`
5: **end if**
6: **repeat**
7: $\bar{u} \leftarrow$ call SAX(`sensor-values[]`)
8: $\delta \leftarrow \|\bar{u} - \bar{r}\|_S$
9: **until** $\delta \leq \theta$
10: call Notify and **goto** line 6

### 3.2 Non-parametric CED

The Non-parametric CED (NPCED) algorithm addresses the requirement of capturing interesting or unusual patterns without having to supply any prior information. This is a distinguishing advantage of CED over composite event techniques which were not designed to cater for unknown thresholds. The procedural steps for NPCED are shown in Algorithm 3.

The basis of NPCED is the ability to train one or more nodes on data known to be normal. This step can take place either online or offline. During the learning phase (lines 2–10), the algorithm determines the rate of change of the monitored process by continuously comparing the string distance of temporally adjacent symbolic representations of sensor values. If the process never changes, the distance between temporally adjacent strings is zero. With a sufficiently long training period, a node can compute the maximum normal change of the process and encode it in a string distance (line 7). Once the learning period is complete, the node makes use of the maximum learnt distance (lines 15–17) to determine whether the process is changing in a manner that differs from expectations, given the training data.

Care must be taken to select a sufficiently long training sample comprised of data corresponding to normal behaviour. For a wide variety of environmental monitoring applications dictated by diurnal cycles, we have found a minimum training period of 24 h to be sufficient. However, in the case where offline data is available a longer training period can be selected as long as the data features in the training set describe normal behaviour. To an extent, the length of the training period is application dependent but through empirical

**Table 1** Sample distance lookup table for 10-letter alphabet

| | a | b | c | d | e | f | g | h | i | k |
|---|---|---|---|---|---|---|---|---|---|---|
| a | 0 | 0 | 0.1936 | 0.5776 | 1.0609 | 1.6384 | 2.3409 | 3.24 | 4.4944 | 6.5536 |
| b | 0 | 0 | 0 | 0.1024 | 0.3481 | 0.7056 | 1.1881 | 1.8496 | 2.8224 | 4.4944 |
| c | 0.1936 | 0 | 0 | 0 | 0.0729 | 0.2704 | 0.5929 | 1.0816 | 1.8496 | 3.24 |
| d | 0.5776 | 0.1024 | 0 | 0 | 0 | 0.0625 | 0.25 | 0.5929 | 1.1881 | 2.3409 |
| e | 1.0609 | 0.3481 | 0.0729 | 0 | 0 | 0 | 0.0625 | 0.2704 | 0.7056 | 1.6384 |
| f | 1.6384 | 0.7056 | 0.2704 | 0.0625 | 0 | 0 | 0 | 0.0729 | 0.3481 | 1.0609 |
| g | 2.3409 | 1.1881 | 0.5929 | 0.25 | 0.0625 | 0 | 0 | 0 | 0.1024 | 0.5776 |
| h | 3.24 | 1.8496 | 1.0816 | 0.5929 | 0.2704 | 0.0729 | 0 | 0 | 0 | 0.1936 |
| i | 4.4944 | 2.8224 | 1.8496 | 1.1881 | 0.7056 | 0.3481 | 0.1024 | 0 | 0 | 0 |
| k | 6.5536 | 4.4944 | 3.24 | 2.3409 | 1.6384 | 1.0609 | 0.5776 | 0.1936 | 0 | 0 |

evaluation we have found that for the majority of applications a 24 h period is sufficient. Some expert knowledge can be useful in determining appropriate training set length for applications with varying characteristics.

The advantage of NPCED is that it offers flexibility to users of a WSN application, in that it does not require prior configuration. In addition, it can potentially lower the energy usage if spatial correlations are exploited in such a way where one node per region takes the responsibility of learning and then communicates the maximum learnt distance to its neighbours.

**Algorithm 3** Non-Parametric Complex Event Detection (NPCED) Algorithm

**Require:** `learnPeriod` $\neq \varepsilon$
1: $max_\delta$, `learnCounter` $\leftarrow 0$
2: **while** `learnCounter` $\leq$ `learnPeriod` **do**
3: $\bar{a} \leftarrow$ call SAX(`sensor-values`$_t$`[]`) {`sensor-values`$_t$ *is a window with the most recent readings*}
4: $\bar{b} \leftarrow$ call SAX(`sensor-values`$_{t-1}$`[]`) {`sensor-values`$_{t-1}$ *is a window temporally shifted by one time unit*}
5: $\delta \leftarrow \|\bar{a} - \bar{b}\|_S$
6: **if** $\delta > max_\delta$ **then**
7: $max_\delta \leftarrow \delta$
8: **end if**
9: Increment `learnCounter` by 1
10: **end while**
11: **loop**
12: $\bar{a} \leftarrow$ call SAX(`sensor-values`$_t$`[]`)
13: $\bar{b} \leftarrow$ call SAX(`sensor-values`$_{t-1}$`[]`)
14: $\delta \leftarrow \|\bar{a} - \bar{b}\|_S$
15: **if** $\delta > max_\delta$ **then**
16: call Notify {*Current distance is greater than max distance learnt*}
17: **end if**
18: **end loop**

### 3.3 Multiple CED

One of the requirements for the CED approach is a provision that makes the method scalable as the number of event subscriptions increase. The advantage of a string representation is that efficient data structures and algorithms from the data mining community can be adapted for the task of Multiple CED (MCED). A common way to achieve this is through *Suffix Trees* [49]: a structure that stores the suffixes of a string in a way that facilitates fast substring searches in $O(m)$ time, where $m$ is the length of the sensor-generated string $\bar{u}$. However, the Suffix Tree is not appropriate for WSNs because the memory requirements for its construction are not predictable at compile-time. Consequently, we selected an equivalent data structure called the *Suffix Array* which is defined as an array of integers in the range 0 to $n-1$, specifying the lexicographic order of the $n$ suffixes of the reference pattern $\bar{r}$ [17].

The reasons we selected the Suffix Array are: first, unlike Suffix Trees, the construction of the array can be performed in a predictable manner known at compile time. This is a desirable property that fits well the extremely resource-constrained WSN execution environment. Second, the Suffix Array is space and time efficient and requires $O(n)$ space where $n$ is the length of the reference pattern and $O(m \log n)$ time [36], where $m$ is the length of the sensor-produced pattern.

The usage scenario for Multiple CED is either the case where the user submits a long reference pattern but is interested in occurrences of smaller subsections of it, or where there are a number of users interested in patterns of arbitrary length. An example of the former can be a user who possesses a week's temperature data and submits it asking to be informed when any sensor-produced pattern matches any subsequence of length greater than $N$. An application context for this could be a data centre air-conditioning unit that operated irregularly before it finally broke at the end of the week. By submitting the week-long pattern and tasking nodes

**Algorithm 4** Multiple Complex Event Detection (MCED) Algorithm

**Require:** `user-patterns[]` $\neq \varepsilon$

**for** $i = 0$ **to** Length(`user-patterns[]`) **do**
  Construct Array for Suffixes of `user-patterns[i]` with suffix length $\geq$ min length of `user-patterns[]`
**end for**
$Suffix Array \leftarrow$ merge Arrays dropping duplicate Suffixes
**loop**
  $\bar{a} \leftarrow$ call SAX(`sensor-values[]`)
  Index $\leftarrow$ call BinarySearch($Suffix Array, \bar{a}$)
  **if** Index $\geq 0$ **then**
    call Notify
  **end if**
**end loop**

to compare their sensed patterns against it, the user can proactively maintain this and other air-conditioning units in the future.

The Suffix Array enables sensor nodes to efficiently determine whether their produced strings match a stored pattern, maintaining the search efficiency as the volume of stored patterns grows. The procedural steps for MCED are outlined in Algorithm 4. Although the algorithm does not show how the Suffix Array can be updated—for instance, if a user submits a new pattern at runtime—this can be achieved using the procedure described in Salson et al. [43]. Finally, the WSN implementation of the Suffix Array relies on insertion sort for the construction and on binary search for the search. In Section 5 we show empirical results that confirm the efficient operation of the Suffix Array for Multiple CED.

### 3.4 Probabilistic CED

With symbolic conversion, there is a process that continuously produces characters from a finite alphabet. We can view this process as a Markov chain where the set of states equals the size of the alphabet.

If the process outputs $x_i$ at time $t$ and then moves to $x_j$ at time $t+1$, the probability for this move is represented by $p_{ij}$ and a state transition from state $i$ to $j$ was observed. The process can also remain in the state it is in with a probability $p_{ii}$, that is when the current character in the string is the same as the previous.

To encode the character transitions, we create and populate a square matrix called the transition matrix. For simplicity, we assume a Markov chain of order one however this is not a strict requirement; higher order (memory) Markov chains can be used depending on the type of data and application at hand.

We then use the Markov model built during the learning phase to predict path probabilities. As with NPCED, the length of the training period must be selected with care; although a 24 h training period will suffice for application with characteristics dictated by diurnal cycles, other applications can benefit from expert knowledge in training period selection. A path probability can be thought of as a realisation of a Markov chain as a path in time through its state space [38]. So a path probability for path $(x_1, x_2 \ldots x_t)$ is given by:

$$P(X_1, X_2 \ldots X_t) = (x_1, x_2 \ldots x_t)$$
$$= P(X_1 = x_1) p_{x_1 x_2} p_{x_2 x_3} \cdots p_{x_{t-1} x_t}$$

Probabilities for strings that are very close to 0 can be flagged as events, as the individual transitions in the string are highly unlikely, given the data segment used to build the transition matrix.

The procedure for Probabilistic CED is listed in Algorithm 5 and is somewhat similar to Non-Parametric CED in that it involves a learning phase. But in contrast to NPCED, the learning phase is used to update the transition matrix according to the character transitions of the current string. Furthermore, a probability threshold is required for detection and is application-dependent. For example, a user may wish to be notified if strings with zero probability are witnessed, that is, strings that were not observed during learning. By

**Algorithm 5** Probabilistic Complex Event Detection (PCED) Algorithm

**Require:** `learnPeriod`, $\theta \neq \varepsilon$

`learnCounter` $\leftarrow 0$
`TransitionMatrix[][]` $\leftarrow 0$
**while** `learnCounter` $\leq$ `learnPeriod` **do**
  $\bar{a} \leftarrow$ call SAX(`sensor-values[]`)
  **for** $i = 0$ **to** Length($\bar{a}$) **do**
    Update `TransitionMatrix[][]` {*Update state transition probabilities*}
  **end for**
  Increment `learnCounter` by 1
**end while**
**loop**
  $\bar{b} \leftarrow$ call SAX(`sensor-values[]`)
  **if** $P(\bar{b}) \leq \theta$ **then**
    call Notify
  **end if**
**end loop**

default, we set the threshold to zero in order to discover previously unseen—during the learning phase—events. Some work will be required for applications to determine an appropriate non-zero value for threshold. One valid method involves introducing a Suffix Tree data structure and annotating paths with probabilities during training, somewhat similar to the technique described in Papadogkonas et al. [41]. The Suffix Tree data structure can be used to obtain the $x$-minimum probability strings observed during training and maximum of those probabilities can be set as the threshold. Since the determination of the threshold is largely application-dependent, we have left the selection of a non-zero threshold to the application user. Line 12 of the algorithm shows the computation of the probability for the current string given the transition matrix populated in training. Similar to Non-parametric CED, Probabilistic CED can also exploit spatial correlations in WSN deployments. For instance a node that has finished learning can share its transition matrix with its neighbours. Since this exchange will only involve a one-off message transmission once the learning phase completes, the communication cost is negligible.

### 3.5 Dynamic sampling frequency management

Dynamic Sampling Frequency Management (DSFM) refers to the ability of WSN nodes in making autonomous decisions about when to increase or reduce their sampling frequency according to the rate of change of the underlying process. The importance of this to a WSN is that it enables nodes to expend less resources during periods of inactivity saving for times of interesting changes. Although it is straightforward for a user to inject a new sampling frequency this assumes undertaking the cost of network communication. The advantage of DSFM is that frequency adjustment decisions can be made locally, involving only computation.

The algorithm for dynamic sampling frequency adjustments is very similar to that of Non-Parametric CED and has been developed for use alongside it, involving an identical training phase. Some care must be taken in the selection of an appropriately long training period—through empirical evaluation we have found that 24 h is sufficient for the majority of applications with characteristics dictated by diurnal cycles. However, expert knowledge can be applied to applications with different requirements. Once learning is complete, the algorithm gradually reduces the sampling frequency until the minimum allowed sampling frequency is achieved. If a distance greater than the maximum distance learnt is witnessed then the sampling frequency is set to the maximum allowed.

The selection of lower and upper bounds for the sampling frequency interval is largely application dependent and should be selected with care, as a naive lower bound selection can increase the number of false negatives (undetected events). In lack of expert knowledge, we have found that setting the lower bound to the sampling frequency value during training phase is appropriate for the applications considered in Section 5. We then set the upper bound to twice the value of the lower bound, select the middle value as the sampling frequency and let the algorithm make appropriate adjustments. Local coordination can greatly help achieve a significantly more appropriate temporal and spatial coverage of a region, effectively minimising the risk of missed events. This can be achieved through coordinated sampling frequency scheduling and time synchronisation. We intend to explore the impact of local coordination further (discussed in Section 6).

Finally, we recognise that some WSN applications can have specific sampling frequency requirements depending of the periodicity of the signal. If these requirements can be specified as a sampling frequency interval, then DSFM can be useful in selecting an appropriate frequency within the interval, relaxing the need for complex pre-deployment signal analysis.

## 4 Integer techniques for efficient implementation

In order to cater for efficient CED we examined the part of the algorithms requiring numeric calculations with floating point types. The lack of Floating Point Unit (FPU) means that all floating point operations are performed in software and therefore are inherently time-consuming.

Using profiling techniques and a custom WSN-specific timing model [56], we refactored the implementation of the CED algorithms in a manner suitable for the constraints of the WSN. Targeting specifically the symbolic conversion routine that consumed the majority of time, we refactored it in an efficient integer-only implementation that reduced the active processing time by more than a factor of 10. This was achieved by a combination of the following actions, listed in order of importance:

– **Integer Programming**. Floating point variables were replaced by their integer equivalents. Functions such as the calculation of standard deviation (cf. [32] for a detailed description of the steps involved in symbolic conversion) were re-written in integer forms.

- **Bitwise Techniques**. Functions such as the square root were replaced by fast integer implementations coded in a manner that utilised bit-level operations. This gave an additional performance boost to the code.
- **General Optimisations**. Unrolling and consolidating loops together with the choice of appropriate variables trimmed off excess milliseconds from the final implementation.

The first target of the CED algorithms was an expensive standardisation operation that was taking an input of numeric sensor values and standardising them by deducting the mean of the sequence and dividing by the standard deviation. Applying standardisation to numeric sensor values accounted for nearly 70% of the algorithm's time—analytic time measurements are presented in Section 5.1. In order to mitigate the high cost of division, we replaced it by a static vector of breakpoints scaled by the standard deviation. The breakpoints are used for the symbolic conversion as a lookup and they are equal to the size of the alphabet. The latter is always smaller than the size of the numeric sensor-produced sequence, therefore scaling the breakpoints involves 10 multiplication operations compared to 40−120 divisions needed in order to standardise the sequence of sensor measurements.

To map the floating point breakpoints to integers we applied binary scaling as an one-off multiplication by a power of 2. In order to operate at numbers of the same scale, we multiplied the intermediate Piecewise Aggregate Approximation (PAA) representation by the same scaling factor—the PAA represents an approximation of the sensor-produced values obtained by dividing the sequence to frames and computing the mean of each frame (cf. [32]). This operation involves a number of multiplications equal to the size of the resulting string.

Bitwise techniques were applied to the calculation of the integer square root, required for computation of standard deviation. Although the integer square root is not strictly required, we found it allows the use of smaller types—a trade off between the square root cost and the use of 64-bit types. From empirical evaluation, we found the cost of using 64-bit types higher compared to 32-bit square root. In addition, we provide an alternative integer square root that does not rely on bitwise operations. It is slightly less efficient but can be used in heterogeneous networks comprising nodes with different byte endianness.

Loop unrolling was applied in a release of the code that accepts fixed number of inputs, for instance a sequence of sensor-produced readings of fixed length. We accept this represents a sacrifice in flexibility and to address it, conditional compilation was provisioned to offer users a selection between variable and fixed sequence length.

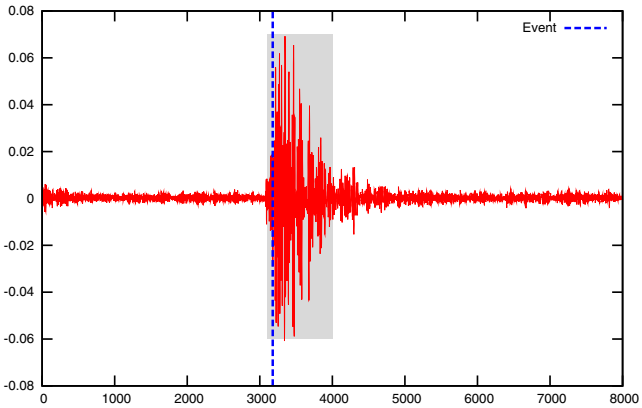
## 5 Evaluation

The performance of the algorithms proposed in this paper has been evaluated against real Wireless Sensor Network data. Whenever possible we have conducted experiments through full implementation on operational WSN systems. In this case, we have employed the Intel WISP platform but results pertaining to performance were collected on TelosB/TMote Sky [42] nodes due to the restricted facilities of WISPs for experimentation. In order to refine the granularity of measurements, we have also opted to simulate the operation of a WSN system by replaying data sets collected in-situ and emulating the operations of a WSN node in software using MATLAB. The latter approach was used with three data sets of special interest which we describe next, and the former to investigate the performance and scalability of the algorithms (Section 5.1).

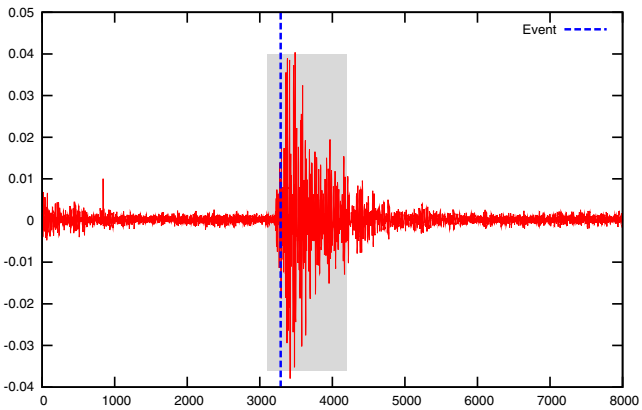
The data sets represent three distinct case studies. First, a volcano-monitoring application with normalised seismometer and acoustic data. This data set contains a large number of organic events and is used for quantitative evaluation of Non-Parametric CED. Second, an experimental indoor network that collected humidity, temperature, light and voltage data. This data set contains a large number of imperfections such as outliers and missing values, making it an ideal candidate for evaluation of all CED algorithms in the presence of faulty observations. Third, ECG and accelerometer data obtained from the UCR Data Mining Archive [24] employed to evaluate the performance against hypothetical pervasive healthcare and context-aware applications.

The objective of all three case studies was to evaluate the accuracy of the CED algorithm across data of varying characteristics. In each case events were identified and their instance of detection was confirmed visually by plotting the detection point against the stream of sensor observations.

*Case study 1: seismic and acoustic data* This series of experiments were carried out on data from the volcanic monitoring deployment of Werner-Allen et al. [50] at Reventador, an active volcano in Ecuador. The deployment comprised 16 sensor nodes that continuously sampled seismic and acoustic data at the relatively high frequency of 100 Hz, for a 19-day period in 2005. The acoustic data was recorded by microphones capturing




(a) Seismic event at node 207 (16/08/2005, 18.30)




(b) Seismic event at node 208 (16/08/2005, 18.30)

**Fig. 5** Examples of non-parametric detection—the *dotted vertical lines* represent the top distances, the *shaded* areas represent the complex event

the infrasonic waves that propagate out of the volcano vent through the atmosphere. The seismic data was recorded by seismometers and captures the waves that may originate from a diffuse zone and may be filtered (scattered and attenuated) within the volcanic edifice.

One of the application goals was to automatically determine when an interesting pattern is sensed. If such a pattern is sensed by a number of nodes, then a base station begins the acquisition of a 60 s segment of data from the network. This mode of operation illustrates the impact of resource constraints to application design: ideally the users need data sensed continuously but the network does not have the resources, in terms of power and bandwidth, to satisfy this need.

For the evaluation of Non-Parametric CED, we exhaustively searched through the entire data set for interesting patterns using compression ratios of 4:1 and 2:1, a ten letter alphabet, and windows of size 32, 64, and 128. The window refers to the subsequence of numeric sensor data passed to the symbolic conversion routine and returned as a string. The algorithm was trained on a subsequence of 1,024 data points that did not contain any events. The computation of distances was calculated by comparing temporally adjacent strings, spaced by four timer ticks. The highest distances were recorded and these represent the onset of an interesting pattern. Figure 5 shows an example of detection, with the dotted vertical lines representing the patterns of interest.

**Table 2** Summary of quantitative detection accuracy results with one and two compression settings

| Compression setting | Detection accuracy | |
|---|---|---|
| Single compression (4 : 1) | Detected: 733 out of 947 | 77.4% |
| Double compression (2 : 1 and 4 : 1) | Detected: 878 out of 947 | 92.7% |

In general, the Non-parametric CED algorithm performed well in the task of identifying interesting patterns and the results are summarised in Table 2. Out of 1, 209 downloaded files, 947 contained clear events while the rest were data containing no interesting patterns. Out of these 947 events, the Non-parametric CED algorithm successfully detected 733 or 77.4%. To further improve detection accuracy, we utilised two different compression ratio settings running in parallel. With this technique, detection accuracy improved to 92.7%.

*Case study 2: indoor deployment* This data set is from an indoor deployment [21] of 54 nodes at the Intel Lab, Berkeley and contains approximately 2.3 million readings of temperature, humidity, light. and voltage sampled at a frequency of 2 Hz. The data contained a small number of real events: an example is the morning spike of 5° in temperature observed by the majority of nodes between 7 and 7.20 am or the sudden decrease of luminosity at midnight. These are potentially attributed to the automated heating system and the nighttime shutdown of the lab, respectively. Synthetic events were manufactured by removing a time series segment and either corrupting by random noise or by selecting values from a uniform distribution in the interval of $[v_{\min}, v_{\max}]$ where $v_{\min}$ and $v_{\max}$ refer to the minimum and maximum values of the entire sensor data set respectively.

A useful finding from this series of experiments was a set of initialisation parameter values that can be used safely for the task of CED. For instance we found that patterns with fewer than 18 data points are difficult to detect consistently. Having tested variations of the input window size ranging from 16 to 836 points, we settle on the value of 40 although larger values can be

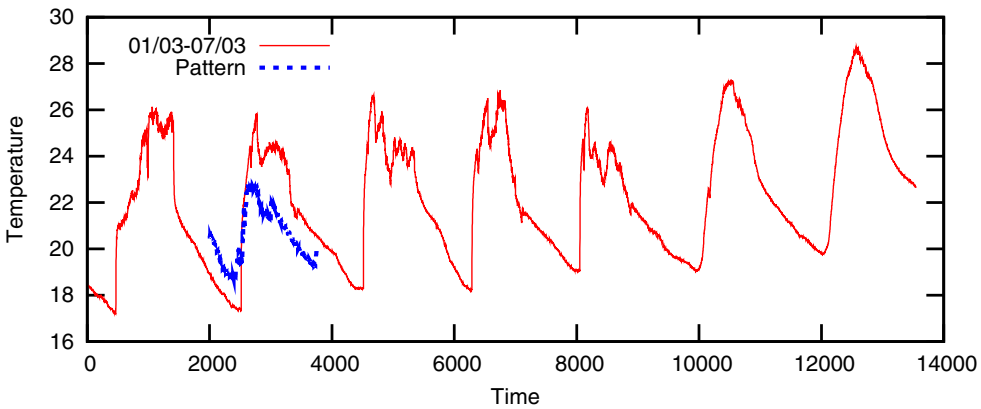
used without hindering detection accuracy. Again, the window size refers to the length of the numeric sensor data that is passed to the symbolic conversion routine. Furthermore, we found that when the pattern is entirely contained in the input size window the accuracy of detection is 100%.

Exact and Approximate CED were evaluated by selecting a data segment, converting it to a string and searching for its occurrence. As an example, a segment of 12 h temperature data was taken, converted to a string and passed to the Exact CED algorithm as a reference pattern to search in a month's data containing the original segment. MATLAB's `Timer` object was employed to emulate the streaming nature of sensor data, generating a single sensor reading every 500 ms.

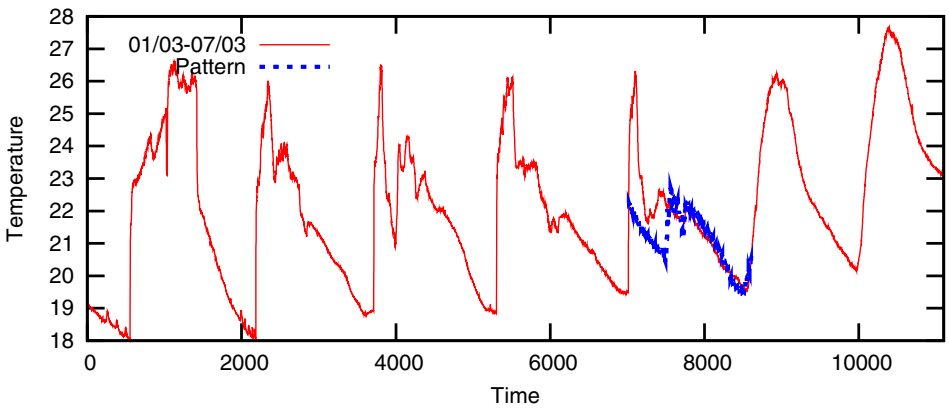
Examples of Approximate CED are shown in Fig. 6a–b, where a search for a specific pattern is performed. Specifically, we are searching in a week's segment of data for a pattern that is not included but the most similar pattern is identified.

*Case study 3: ECG and accelerometer data* This experiment evaluates the Non-Parametric CED algorithm on ECG and accelerometer data obtained from the UCR Data Mining archive [24]. For the former, we tested two cases: the first is a normal ECG that turns to Super Ventricular and the second is normal turning to Malignant Ventricular. The detection was accurate in both cases and the algorithm managed to pinpoint the change with reasonable latency. An example is shown in Fig. 7 with the detected points of change identified and marked as appropriate.

In addition, an experiment of Non-parametric CED was conducted on 3-axis accelerometer data obtained from a Sony ERS-210 Aibo Robot. In this case, the accelerometer data was sampled at a rate of 125 Hz. In terms of coordinate frames of the robot, the positive $X$ axis points towards the front of the robot, positive $Y$ points out of the left side of the robot, and positive $Z$ points straight up. The subset of the data tested is a segment sampled when the robot was playing. We used a quarter of the data to train the algorithm and



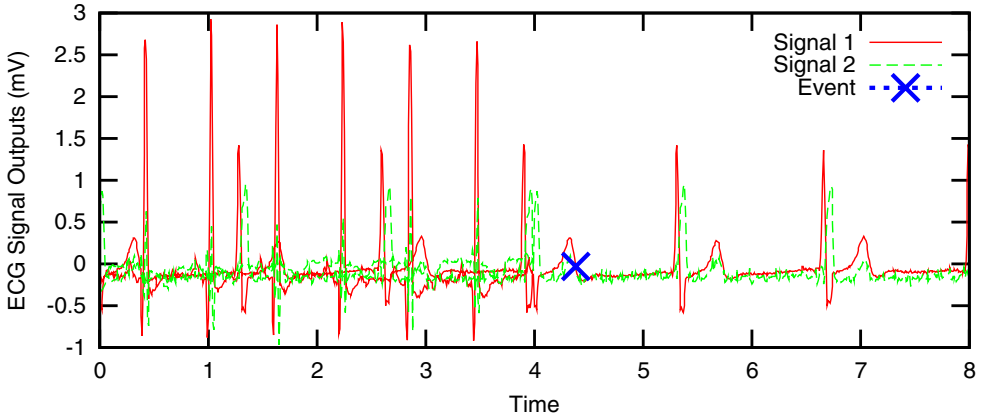

(a) The pattern is temperature data for the dates 19/03-20/03 (Node1). The algorithm identifies the closest match to the pattern



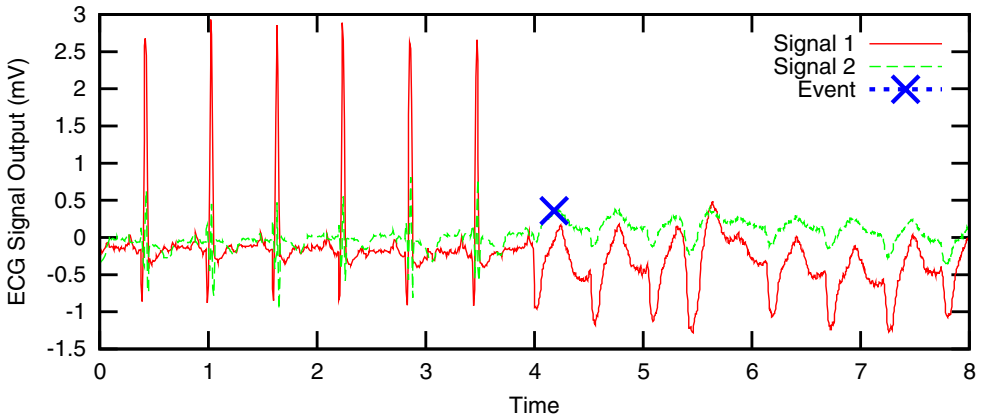

(b) Searching for a pattern (18/03-19/03) in the week 01/03-07/03 (Node4). The closest match is identified at point 7,017

**Fig. 6** Examples of approximate matching

**Fig. 7** ECG—two different datasets that start normal but change at 4 s

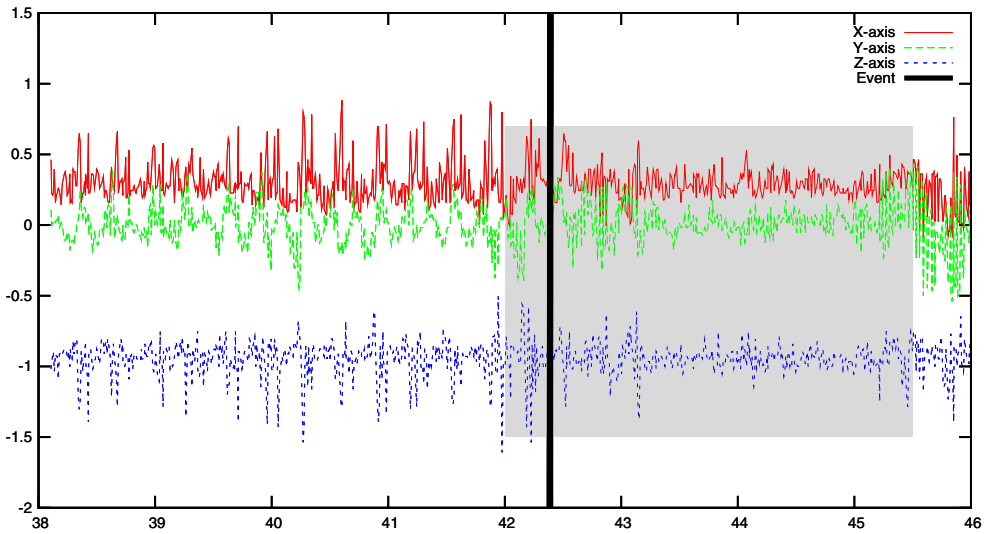



(a) ECG–normal turning to Supra-Ventricular. The exact change happens at precisely 4 seconds. The event is identified at 4.38




(b) ECG–normal turning to Malignant Ventricular. The exact change happens at precisely 4 seconds. The event is identified a t4.18

**Fig. 8** CED in 3-axis accelerometer readings from a robot playing. The most unusual point is identified by the algorithm at time 42.38



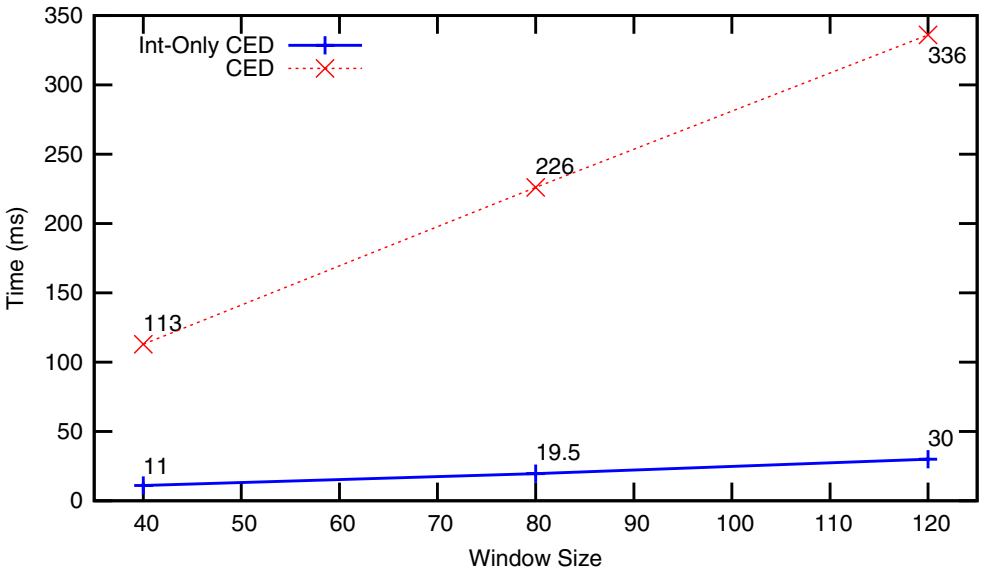



**Fig. 9** Comparison of running times for two alternative implementations of the CED algorithm on a TMote Sky/TelosB. *CED* refers to an implementation that relies heavily on a number of floating point operations. *Int-Only CED* refers to an optimised implementation that has been refactored using integer programming

subsequently identified point 536 (Fig. 8) as the point of highest distance.

We intentionally selected data from sensors of eight different modalities to empirically evaluate the performance of our suggested approach against a number of hypothetical application domains. Further to that, a WSN-implemented demonstration testbed was designed and presented where one or more sensor nodes train using light readings in a room. Complex events are induced using variations of the brightness caused by a flash light over the light sensor. We were able to demonstrate successful real-time CED running on WSN nodes. Further experiments to the same effect with different sensor data such as accelerometer, acoustic and so on were performed using a multi-sensor [13] plug in.

### 5.1 Experiments and performance analysis

We now focus on the performance of the CED algorithms which establishes the suitability for the target platform. First, we consider the impact of the Integer optimisations of Section 4: to evaluate the performance of the integer-only CED algorithm, we measured its execution using the `Timer` component of TinyOS and specifically the `startOneShot` and `stop` commands, at the beginning and end of the function call to CED respectively. The times reported were collected on a TMote Sky/TelosB [42] node. The results of integer-only compared to classic CED, are shown in Fig. 9. To highlight the energy saving potential, assuming a 1Hz sampling frequency, the current consumption for a typical sensor-produced sequence of length 40 is 73.7 mA for the integer implementation compared to 251.74 mA for the implementation involving floating point numbers.

Results for individual operations are shown in Table 3, and although RAM usage is slightly higher for the integer-only implementation it still leaves plenty of RAM for other WSN-specific tasks (the total RAM is 10 Kb). The 10-fold reduction in time needed for CED represents a significant gain and is further proof that CED algorithms are not only accurate and feasible but also very efficient.

Second, we evaluated the scalability aspect of the Multiple CED algorithm that utilises the Suffix Array

**Table 3** Comparison of CED to integer-only implementation

| Operation | CED | | Int-Only CED | |
|---|---|---|---|---|
| S2. Standardise | | | | |
| a. Mean | 12 ms | (10.62%) | | |
| b. Std dev | 42 ms | (37.17%) | 5,958 µs | (54.17%) |
| c. Subtract & divide | 25 ms | (22.12%) | | |
| S3. Get PAA | 24 ms | (21.24%) | 4,125 µs | (37.5%) |
| S4. Get Symbols | 10 ms | (8.85%) | 916 µs | (8.33%) |
| Total time | 113 ms | | 11 ms | |
| RAM image size (bytes) | 766 | | 1,180 | |

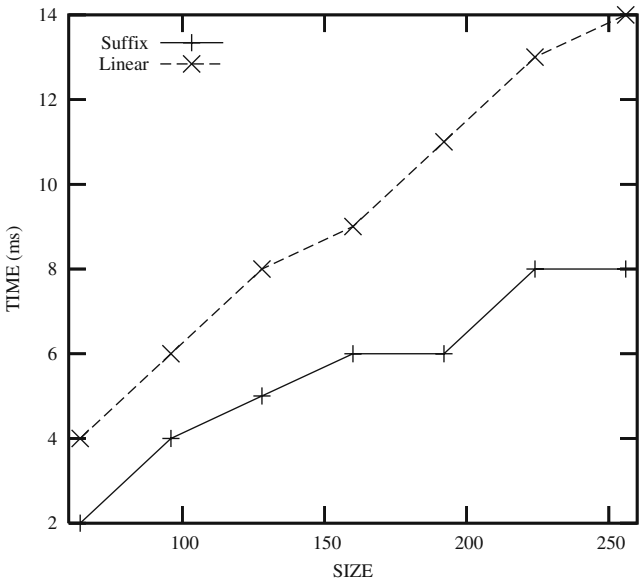



**Fig. 10** Comparison of suffix array binary search against linear search. The SIZE attribute refers to the cumulative size of the concatenated patterns (each pattern representing a complex event)

data structure on real WSN nodes. This series of experiments profiles the behaviour of the algorithm as the number of user-submitted events increase. In order to provide a basis for evaluation, we implemented a Linear Search algorithm as an alternative that determines whether a sensor-produced pattern exists in a collection of user-submitted reference patterns. Multiple user patterns were concatenated with each pattern terminated by appending $ in order to avoid spurious matches. In this manner, Linear Search was used as the benchmark for comparison to Suffix Array and Binary Search.

The results of a comparison between binary searching the Suffix Array and performing a linear search are shown in Fig. 10. The *Size* attribute refers to the cumulative size of the concatenated user-submitted patterns, and the *Time* attribute refers to the time it took a WSN node to perform each type of search. For instance a node needs just 8 ms to simultaneously monitor 16 distinct event occurrences each with a string length of 16 characters. These results are also shown in Table 4: as the size grows, linear search becomes increasingly slower. None of the test cases represent a worst-case for the linear search. The worst-case is when the pattern is not found at all in the user-submitted reference pattern. Further to our WSN-specific search results, there are plenty more performance statistics comparing searches using alternative data structures, for example [36] presents a comprehensive comparison of suffix trees and Suffix Arrays.

These results confirm empirically that the algorithms proposed are both feasible for execution in the WSN platform and scalable as the number and size of user-submitted reference patterns grow. Last, the robustness of the algorithms has been verified by executing continuously on an indoor testbed of four unattended nodes deployed in one of our laboratory buildings for the period of one month during which they processed over one million temperature readings. The nodes were running the Non-Parametric CED algorithm and used the first day of deployment as training period. The maximum distance learnt during training was used to determine occurrence of unusual patterns and the nodes converted numeric sequences of 40 readings to strings using a 4:1 compression ratio and a ten letter alphabet. As expected, no Complex Events were detected and the maximum distance observed during training was not exceeded by the comparison of temporally adjacent sequences. There were no abnormal events such as heating failures during the month of operation, so the algorithm operated correctly without falsely notifying. Finally, outlier readings observed during the month did not affect the performance or the accuracy of the algorithm.

## 6 Future work

We recognise that certain research questions posed in this article are only partially addressed. To this end, we intend to further investigate work in the following three directions:

– In Section 3.2, it was mentioned that Non-parametric CED can incorporate local coordination to balance energy requirements across a local group of nodes, for instance by sharing the maximum learnt distance with neighbours or by alternating learning between nodes. The potential benefit

**Table 4** Performance times (in ms) for suffix search compared to linear search

| | Size of concatenated patterns | | | | | | |
|---|---|---|---|---|---|---|---|
| | 64 | 96 | 128 | 160 | 192 | 224 | 256 |
| Times (ms)—suffix array | 2 | 4 | 5 | 6 | 6 | 8 | 8 |
| Times (ms)—linear search | 4 | 6 | 8 | 9 | 11 | 13 | 14 |

of coordination to Non-parametric and Probabilistic CED can be investigated further. Partitioning a WSN into groups or cells with each group responsible for a geographic region or a spatial entity can provide some insight and quantification on the trade offs between communication overhead and distribution of computation.

- Given the wide uses of WSNs and the varying characteristics of the collected data, we recognise that in certain cases the CED algorithms can be augmented if running in parallel with other detection techniques. Similar to *Ensemble methods* [39] where multiple models are employed to improve predictive performance, we believe that multiple detection algorithms can also improve accuracy. The counter-argument is that a collection of detectors will inevitably require more resources and in the future we intend to investigate this trade off further.
- A deliverable with practical importance to WSN researchers, is a collection of integer-only implementations of a family of functions, such as `sqrt`, `sin`, `cos`, `tan` involving floating point numbers. A software library with fixed-point implementation of varying precision for these functions can be a valuable tool to applications that perform target tracking or data analysis. As discussed in Section 4, CED relies only on square root and an integer version has already been implemented, however extending the implementation to functions outside the scope of CED can benefit the wider WSN programming community.

Addressing the above directions through evaluation on real WSNs will extend the work on CED and benefit users of reactive applications across a number of domains.

## 7 Discussion & conclusions

We have presented a comprehensive family of algorithms for Complex Event Detection (CED) as a valid alternative to the composite event model. The CED algorithms offer several advantages including the ability to search for patterns difficult or even impossible to capture using traditional composite techniques and thresholds. The lightweight nature of the algorithms make them suitable for a number of extremely resource-constrained platforms without compromising detection accuracy or incurring radio communication costs. With respect to the latter, the CED algorithms contribute to energy savings by promoting reliable reactive alerting—a superior solution to costly passive monitoring and out-of-network processing.

The evaluation findings contribute towards WSN-related research in the field of reactive systems. As far as we were able to confirm from literature, this is the first work addressing the complex nature of WSN events that are common across a number of applications. WSN applications usually implement custom solutions fit for a specific data or deployment but lacking a generic event model, similar to that found in conventional database systems. Our algorithms, methods and techniques deliver a mechanism where Complex Events can be defined and submitted to the WSN easily while detection can be performed according to a selection of methods to suit different application needs. The nature of CED guarantees a fixed, efficient execution cost and does not restrict users in expressing interests in events as the composite event model, where valid expressions can have arbitrary length and therefore become unsuitable for execution. The efficient nature of the algorithms requiring very few resources, complements the generic goal of prolonged lifetime and makes them fit for extremely resource-constrained devices. Moreover, the spatial aspect of Complex Events needs to be considered and we have started work [54] towards this direction that addresses location estimation of events with position and direction in physical space.

Finally, we show that there exist a solution to the Complex Event Detection problem in WSNs that is both efficient and appropriate for the data characteristics of WSNs. This solution does not require provision of new communication paradigms and has been shown to be compatible with existing state-of-the-art Pub/Sub, without adding a communication overhead or modifications to underlying protocols. The efficiency of the solution has the consequence of low resource and power requirements, making it a superior alternative to threshold-based and composite event detection techniques.

## References

1. Arzen KE, Bicchi A, Dini G, Hailes S, Johansson KH, Lygeros J, Tzes A (2007) A component-based approach to the design of networked control systems. Eur J Control 13(2–3):261–279
2. Basha EA, Ravela S, Rus D (2008) Model-based monitoring for early warning flood detection. In: SenSys '08: proceedings of the 6th ACM conference on embedded network sensor systems, pp 295–308
3. Buettner M, Prasad R, Sample A, Yeager D, Greenstein B, Smith J, Wetherall D (2008) RFID sensor networks with the intel WISP. In: SenSys. ACM, pp 393–394

4. Burrell J, Brooke T, Beckwith R (2004) Vineyard Computing: Sensor Networks in Agricultural Production. IEEE Pervasive Computing 3(1):38–45
5. Cardell-Oliver R, Kranz M, Smettem K, Mayer K (2005) A reactivetion soil moisture sensor network: design and field evaluation. IJDSN 1(2):149–162
6. Casey K, Lim A, Dozier G (2008) A Sensor Network Architecture for Tsunami Detection and Response. IJDSN 4(1):28–43
7. Ceriotti M, Mottola L, Picco GP, Murphy A, Stefan G, Corrà M, Pozzi M, Zonta D, Zanon P (2009) Monitoring heritage buuldings with wireless sensor networks: the Torre Aquila deployment. In: IPSN '09: proceedings of the 2009 international conference on information processing in sensor networks. IEEE Computer Society, Washington, pp 277–288
8. Chakravarthy S, Krishnaprasad V, Anwar E, Kim SK (1994) Composite events for active databases: Semantics, contexts and detection. In: Proceedings of the international conference on very large data bases, pp 606–606
9. Chin JC, Yau DKY, Rao NSV, Yang Y, Ma CYT, Shankar M (2008) Accurate localization of low-level radioactive source under noise and measurement errors. In: SenSys '08: proceedings of the 6th ACM conference on embedded network sensor systems, pp 183–196
10. Desnoyers P, Ganesan D, Li H, Li M, Shenoy P (2005) PRESTO: A predictive storage architecture for sensor networks. In: Tenth workshop on hot topics in operating systems (HotOS X)
11. Doolina D, Sitar N (2004) Wireless sensors for wildfire monitoring. In: SPIE symposium on smart structures & materials
12. Doraiswamy PC, Moulin S, Cook PW, Stern A (2003) Crop yield assessment from remote sensing. Photogramm Eng Remote Sensing 69(6):665–674
13. EasySen C (2008) SBT80—multi modality sensor board. http://www.easysen.com/SBT80.htm. Accessed 1 August 2009
14. Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. SIGMOD Rec 23(2):419–429. doi:10.1145/191843.191925
15. Gao T, Greenspan D, Welsh M, Juang R, Alm A (2005) Vital signs monitoring and patient tracking over a wireless network. In: Proceedings of the 27th annual international conference of the IEEE EMBS, pp 102–105
16. Gunatilaka A, Ristic B, Gailis R (1988) Radiological source localisation. DSTO-TR-1988. http://hdl.handle.net/1947/8682. Accessed July 2007
17. Gusfield D (1997) Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge University Press
18. Han Q, Jayasumana AP, Illangaskare TH, Sakaki T (2008) A wireless sensor network based closed-loop system for subsurface contaminant plume monitoring. In: Proceedings of 24th IEEE international parallel and distributed processing symposium. IEEE, pp 1–5
19. Hauer JH, Handziski V, Köpke A, Willig A, Wolisz A (2008) A component framework for content-based publish/subscribe in sensor networks. In: EWSN. Springer, pp 369–385
20. Huang L, Garofalakis M, Hellerstein J, Joseph A, Taft N (2006) Toward sophisticated detection with distributed triggers. In: MineNet '06: proceedings of the 2006 SIGCOMM workshop on mining network data, pp 311–316
21. Intel (2004) Lab data (Berkeley). http://db.csail.mit.edu/labdata/labdata.html. Accessed 2 June 2004
22. Ishida H, Nakamoto T, Moriizumi T (1998) Remote sensing of gas/odor source location and concentration distribution using mobile system. Sens Actuators, B, Chem 1–2(49):52–57
23. Katsiri E, Ho M, Wang L, Lo B, Toumazou C (2007) Embedded real-time heart variability analysis. In: 4th international workshop on wearable and implantable body sensor networks (BSN 2007), pp 128–132
24. Keogh E (2008) The UCR time series data mining archive. http://www.cs.ucr.edu/eamonn/TSDMA/. Accessed December 2008
25. Keogh E, Lonardi S, Ratanamahatana CA (2004) Towards parameter-free data mining. In: KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York, pp 206–215
26. Keogh E, Lin J, Fu A (2005) HOT SAX: efficiently finding the most unusual time series subsequence. IEEE international conference on data mining, pp 226–233
27. Kim S, Pakzad S, Culler D, Demmel J, Fenves G, Glaser S, Turon M (2007) Health monitoring of civil infrastructures using wireless sensor networks. In: IPSN '07: proceedings of the 6th international conference on information processing in sensor networks, pp 254–263
28. Kirsch C, Mattingley-Scott M, Muszynski C, Schaefer F, Weiss C (2007) Monitoring chronically ill patients using mobile technologies. IBM Syst J 46(1):85–93
29. Krishnamurthy L, Adler R, Buonadonna P, Chhabra J, Flanigan M, Kushalnagar N, Nachman L, Yarvis M (2005) Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In: SenSys '05: proceedings of the 3rd international conference on embedded networked sensor systems, pp 64–75
30. Levis P, Culler D (2002) Maté: a tiny virtual machine for sensor networks. In: ASPLOS-X: proceedings of the 10th international conference on architectural support for programming languages and operating systems. ACM, New York, pp 85–95
31. Levis P, Brewer E, Culler D, Gay D, Madden S, Patel N, Polastre J, Shenker S, Szewczyk R, Woo A (2008) The emergence of a networking primitive in wireless sensor networks. Commun ACM 51(7):99–106
32. Lin J, Keogh E, Lonardi S, Chiu B (2003a) A symbolic representation of time series, with implications for streaming algorithms. In: DMKD '03: proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery. ACM, New York, pp 2–11
33. Lin M, Kumar A, Qing X, Beard SJ, Russell SS, Walker JL, Delay TK (2003b) Monitoring the integrity of filament-wound structures using built-in sensor networks. In: Society of photo-optical instrumentation engineers (SPIE) conference series, vol 5054, pp 222–229
34. Lo BPL, Thiemjarus S, King R, Yang Gz (2005) Body sensor network—a wireless sensor platform for pervasive healthcare monitoring. In: Adjunct proceedings of the 3rd international conference on pervasive computing (PERVASIVE'05, pp 77–80
35. Malan D, Fulford-jones T, Welsh M, Moulton S (2004) CodeBlue: an ad hoc sensor network infrastructure for emergency medical care. In: International workshop on wearable and implantable body sensor networks
36. Manber U, Myers G (1990) Suffix arrays: a new method for on-line string searches. In: SODA '90: proceedings of the first annual ACM-SIAM symposium on discrete algorithms, society for industrial and applied mathematics. Philadelphia, pp 319–327
37. Marshall IW, Price MC, Li H, Boyd N, Boult S (2007) Multisensor cross correlation for alarm generation in a deployed sensor network. In: EuroSSC. Springer, pp 286–299

38. Norris JR (1997) Markov chains. Cambridge University Press
39. Opitz D, Maclin R (1999) Popular ensemble methods: an empirical study. J Artif Intell Res 11(169–198):12
40. Paek J, Chintalapudi K, Govindan R, Caffrey J, Masri S (2005) A wireless sensor network for structural health monitoring: performance and experience. In: EmNets '05: proceedings of the 2nd IEEE workshop on embedded networked sensors. IEEE Computer Society, Washington, pp 1–9
41. Papadogkonas D, Roussos G, Levene M (2006) Discovery and ranking of significant trails. In: Second international workshop on exploiting context histories in smart environments (ECHISE) at Ubicomp, p 8
42. Polastre J, Szewczyk R, Culler D (2005) Telos: enabling ultra-low power wireless research, pp 364–369
43. Salson M, Lecroq T, Léonard M, Mouchard L (2009) Dynamic extended suffix arrays. J Discret Algorithms
44. Shin M, Tsang P, Kotz D, Cornelius C (2009) DEAMON: energy-efficient sensor monitoring. In: Sensor, mesh and ad hoc communications and networks, 2009. SECON '09. 6th Annual conference on IEEE communications society, pp 1–9
45. Steere DC, Baptista A, McNamee D, Pu C, Walpole J (2000) Research challenges in environmental observation and forecasting systems. In: MOBICOM, pp 292–299
46. Stiefmeier T, Roggen D, Tröster G (2007) Gestures are strings: efficient online gesture spotting and classification using string matching. In: BodyNets '07: proceedings of the ICST 2nd international conference on body area networks, pp 1–8
47. Stoianov I, Nachman L, Madden S, Tokmouline T (2007) PIPENETa wireless sensor network for pipeline monitoring. In: IPSN '07: proceedings of the 6th international conference on information processing in sensor networks, pp 264–273
48. Talzi I, Hasler A, Gruber S, Tschudin CF (2007) PermaSense: investigating permafrost with a WSN in the Swiss alps. In: EmNets. ACM, pp 8–12
49. Ukkonen E (1995) On-line construction of suffix trees. Algorithmica 14(3):249–260
50. Werner-Allen G, Lorincz K, Johnson J, Lees J, Welsh M (2006) Fidelity and yield in a volcano monitoring sensor network. In: OSDI, USENIX association, pp 381–396
51. Werner-Allen G, Dawson-Haggerty S, Welsh M (2008) Lance: optimizing high-resolution signal collection in wireless sensor networks. In: Proceedings of the 6th ACM conference on embedded network sensor systems, pp 169–182
52. Xue W, Luo Q, Chen L, Liu Y (2006) Contour map matching for event detection in sensor networks. In: SIGMOD '06: proceedings of the 2006 ACM SIGMOD international conference on management of data, pp 145–156
53. Zoumboulakis M, Roussos G (2009a) Efficient pattern detection in extremely resource-constrained devices. In: Sixth annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks
54. Zoumboulakis M, Roussos G (2009b) Estimation of pollutant-emitting point-sources using resource-constrained sensor networks. In: GSN '09: proceedings of the 3rd international conference on geosensor networks, pp 21–30
55. Zoumboulakis M, Roussos G (2009c) In-network pattern detection on Intel WISPs. (Demo Abstract). In: Proceedings of wireless sensing showcase
56. Zoumboulakis M, Roussos G (2009d) Integer-based optimisations for resource-constrained sensor platforms. In: First international ICST conference, S-CUBE 2009, pp 144–157