Escalation: Complex Event Detection in Wireless Sensor Networks

Michael Zoumboulakis and George Roussos

School of Computer Science and Information Systems,
Birkbeck College, University of London
Malet Street, London WC1E 7HX, UK
{mz,gr}@dcs.bbk.ac.uk

Abstract. We present a new approach for the detection of complex events in Wireless Sensor Networks. Complex events are sets of data points that correspond to interesting or unusual patterns in the underlying phenomenon that the network monitors. Our approach is inspired from time-series data mining techniques and transforms a stream of real-valued sensor readings into a symbolic representation. Complex event detection is then performed using distance metrics, allowing us to detect events that are difficult or even impossible to describe using traditional declarative SQL-like languages and thresholds. We have tested our approach with four distinct data sets and the experimental results were encouraging in all cases. We have implemented our approach for the TinyOS and Contiki Operating Systems, for the Sky mote platform.

Keywords: Event Detection, Complex Events, Parameter-free Detection, Data Compression, Network Control, Reactive Sensor Networks.

1 Introduction

Our approach is aimed at the extremely resource-constrained devices known as *motes*; motes combine a radio component, a microcontroller, an array of sensors and an energy source (e.g. battery). They are designed for operating unattended for long periods of time (typically for as long as their energy source lasts). Our target platform for which we are developing code is the TMote Sky motes [1].

The event-driven model in sensor networks broadly prescribes that individual or groups of sensor nodes should react programmatically to events, very much in the same manner a traditional DBMS reacts to violating transactions. The database abstraction for sensor networks offers the capability of querying with the use of a declarative SQL-like language and specifying reactive functionality with the use of triggers and predicates. The problem arises from the fact that the use of predicates and thresholding with relational and logic operators, are not expressive enough to capture most of the complex events that occur in sensor networks. Complex events are sets of data points that constitute a pattern; this pattern shows that something interesting or unusual is occurring in the underlying process monitored. While viewing the sensor network as a database

G. Kortuem et al. (Eds.): EuroSSC 2007, LNCS 4793, pp. 270–285, 2007.

is useful for running queries over streaming data, it offers little help in detecting interesting patterns of complex events.

A separate problem arises when users do not know in advance the event type they are looking for. A benefit of our approach is that it also allows non-parametric event detection. In this sense, motes become context-aware and they are capable of detecting relative change. Practically, this is implemented with a concept borrowed from Machine Learning: motes go through a learning phase that is known to be normal. During this phase, they continuously compare strings (that correspond to temporally adjacent sets of readings) and compute distances among them. Once the learning phase is complete, these distances are used for non-parametric detection — the distances effectively constitute the normal context; a distance never seen before represents some unseen and new change.

Another mode of operation offered by our approach is approximate event detection: users can search for events that are *like* other events by submitting an event template. Practically, the template (that is a set of real-valued readings) gets translated into a string representation and submitted to the relevant motes. Streaming sensor readings are then transformed to strings and comparisons are performed against the template. A distance (between the template and the newly-generated string representations of streaming readings) that approaches zero signifies that the reporting mote is experiencing a very similar event to the one submitted. Of course a zero distance denotes that the exact same event has occurred.

Perhaps the strongest point of our approach is that it makes motes context-aware: since they can sense relative change they can adjust the sampling frequency dynamically and prolong their lifetime by conserving power resources. Furthermore, optimisation of sleep scheduling can ensure that in periods of low activity the majority of the nodes are asleep. If distances start to increase, then explicit wake-up messages can be sent to sleeping nodes tasking them with sensing, processing or sending.

In the sections to follow, we will discuss in great detail our approach and we will provide a high-level description of our algorithm. We will present results from our experiments with sensor data sets, and we will conclude by outlining future plans (including an indoor test deployment).

1.1 Target Applications

Our approach can be beneficial to a fairly large selection of applications, namely:

- *Pervasive Health Monitoring.* One aspect of this has to do with monitoring the activity of human organs over time (such as ECG, EEG, etc.) and projects such as [2]. An example of a Complex Event is a heart attack — if there is any pattern in the signal just before the heart attack. then a detected event can help save lives. The other aspect has to do with Pervasive Healthcare Monitoring [3,4]: monitoring patients at their sensor-equipped homes rather than hospitals. Any "abnormal" behaviour can flag an event — for instance an elderly person not getting up at their normal time may indicate

illness. The system can react to this by informing a social worker to visit or call. We are also testing our approach to a patient classification problem using the same distance metric for similarity / dissimilarity. The data comes from patients suffering from Cystic Fibrosis and healthy controls. Data has been collected using the Cyranose 320 electronic nose — a device with 32 sensors that can be used for bacterial classification [5].
- *Environmental (Outdoor or Indoor) Monitoring.* Normally, this involves monitoring physical attributes of the environment such as temperature, humidity, light, etc. Examples include: meteorological stations, wildfire monitoring (and projects such as the Firebug [6,7], indoor monitoring (projects such as the Intel Lab Deployment [8] of which data we have used for our experiments).
- *Athlete Performance Monitoring.* Approaches like the SESAME project [9] use motion sensors (such as accelerometers) in order to monitor attributes related to the performace of elite athletes. A Complex Event in this case can be a pattern that denotes a change in performance — for instance a sub-optimal start to a sprint.

To generalise the above, our approach can benefit any system or process that exhibits long periods of normality and relatively much smaller periods of rare events.

2 Conversion of Streaming Sensor Data to a Symbolic Representation

For the conversion to string we use the Symbolic Aggregate Approximation (SAX) algorithm [10,11] which is a very mature and robust solution for mining time-series data. SAX creates an approximation of the original data by reducing its original size while keeping the essential features — this fact makes it a good choice for the sensor network setting because it offers the advantage of data *numerosity reduction* which is a very desirable property since it can contribute to saving power (sensors transmit a string which is a reduced approximation of the original data set).

The distance between SAX-representations of data has the desirable property of lower-bounding the Euclidean distance. Lower-bounding means that for two time-series sets Q and C there exist symbolic representations $\grave{Q}$ and $\grave{C}$ respectively such that $D_{LB}(\grave{Q}, \grave{C}) \leq D(Q, C)$ where $D(Q, C)$ is the Euclidean distance.

Once the time series has been converted to a symbolic representation, other distance metrics & algorithms can be applied for measuring similarity as well as for pattern-matching. We have tested a variety of metrics such as the Hamming Distance, Levenshtein Distance, Local & Global Alignment, Kullback-Leibler Divergence and Kolmogorov Complexity (as described in [12]).

3 *Escalation*: Applying SAX to Streaming Sensor Data

We have combined SAX and the distance metric used by SAX with an algorithm that detects complex events without the need of the user supplying an explicit

threshold. The algorithm learns what constitutes an interesting change, very much in the machine learning sense. The main beenfits of introducing such an approach to sensor networks are:

- *Dynamic Sampling Frequency Management.* This refers to the ability to autonomously make a decision on whether to increase or decrease the sampling frequency. The decision is based upon the rate of change.
- *Parameter Free Detection.* This refers to event-detection without supplying thresholds or generally an event description.
- *Successful Detection of Complex Events.* This refers to the ability to detect complex events that are impossible to describe or capture using traditional techniques such as triggers and thresholding.

Firstly, longevity in a sensor network is of crucial importance, so by monitoring how quickly attribute values change a sensor node can make an autonomous decision to increase or decrease the sampling frequency accordingly. Our algorithm essentially compares a symbolic representation of a current set of values to a previous one and calculates the distance. If the distance is zero or a very small number then the sampling frequency can be reduced. Conversely, sampling frequency can be doubled as soon as the distance starts to increase and moreover local broadcasts can be made to neighbouring nodes in order to issue wake-up calls or increase sampling frequency commands. This way the sleep schedule of nodes can be naturally adapted to the rate of the physical change. This advantage is important because it can significantly prolong the useful lifetime of the sensor network. In addition it means that sleep scheduling can be optimised and the biggest percentage of the network can be asleep for the majority of the time. If, for example, the network is split into regions, then only two nodes per region need to be awake at any given time. The second node acts as a verifier or a fail-over for the first node.

Secondly, specifying a threshold based on user experience or expectancy can be an arbitrary process. A small threshold means that the nodes are overworking and similarly a high threshold may mean that interesting changes are lost. Moreover, threshold are susceptible to outliers. Thresholding is also fairly fixed in nature; the threshold is either compiled into a binary image at pre-deployment or it is injected as a VM command at runtime. Either way it involves human intervention to the system. Our algorithm offers complex detection by allowing the user to either specify the pattern they are looking for and then search for proximity to that pattern (in terms of approximate string matching) or having the node make an autonomous decision on what is interesting based on monitoring some normal data. This second advantage allows us to successfully detect a whole class of events with no programming effort — all the user will have to do is to press start (or to be precise compile the binary image into the node) and the node will autonomously do the rest. There is no need for specifying what events to look for by using cumbersome and error-prone code. The node will take care of learning normality and then deducing what is not normal.

Lastly, we have mentioned earlier that events in sensor networks are conceptually different than events in conventional DBMS: in sensor networks we deal

with signals or streams of continuous real-time data. We do not deal with transactions. So a complex event in that sense is a set of data points that satisfies some condition e.g. this condition could be a state change. Or to use an alternative description, a complex event is a pattern that its frequency of occurrence is greatly different from that which we expected, given previous experience or observations [11]. By using SAX with the sensor network data, we are able to detect such complex patterns with good accuracy.

Algorithm 1. Complex Event Detection Algorithm

```
 1: variables counter, bufferSize, distanceBufferSize, saxWindow, stillLearning;
 2: call initialise (set values to the above, set stillLearning to true)
 3: while (stillLearning is true) do
 4:    get data from sensors;
 5:    if counter ≤ bufferSize then
 6:       fill buffer with data; break;
 7:    else
 8:       update buffer with new reading and shift elements;
 9:       extract two subsections from buffer (say, a and b);
10:       call SAX passing a and b and update distances (if dist(a,b) ≠ 0);
11:       if distance buffer is fully populated then
12:          stillLearning = false;
13:          select maximum distance observed during learning phase;
14:       end if
15:    end if
16: end while
17: while true do
18:    get data from sensors;
19:    update buffer with new reading and shift elements;
20:    extract two subsections from buffer (say, a and b);
21:    call SAX passing a and b and update distances (if dist(a,b) ≠ 0);
22:    if dist(a,b) ≥ maximum distance set in learning phase (line 13) then
23:       report distance and counter (representing timestamp) or do some other action
24:    end if
25: end while
```

3.1 High-Level Description

Let us start by describing the algorithm is pseudocode. We have developed 2-3 variants of this, but the basic idea is the same: namely, monitor distances during learning phase, choose minimum, maximum or average distance, and then compare adjacent string representations. If their distance is greater than the maximum distance observed during training, then it is highly likely that an event is occurring.

There is a further variation of the algorithm that offers the escalation functionality, in that it calls a third optional phase to investigate a reported complex event even further. This is implemented by a function call on line 23 (not shown).

The whole of the readings set is passed to this function, and the subsections are expanded in order to distinguish true from false positives. This way, the sensitivity (high true positive rate) and selectivity (low false positive rate) of the algorithm are maximised. Empirically, from running our experiments we notice that an escalation phase is not always necessary as in most cases the algorithm will detect the exact point of the event with great accuracy.

Summarising, there are two main phases to the algorithm, with an optional third;

- *Learning Phase*, where the algorithm keeps track (learns) of the distances it has seen.
- *Initial Detection Phase* during which SAX is called to convert adjacent sets of numeric readings to strings and calculate their distance. An event is flagged if the distance between two strings is greater that the maximum distance observed during learning.
- *Escalation Phase* is a temporally deferred operation that varies the window size over the numeric sets and produces progressively larger strings. It is used to increase the accuracy of the algorithm (by pruning false positives).

Essentially, the algorithm puts the sensor in an infinite loop that forces them to convert readings to strings, compute their distance and test it against a distance learned during the initial learning phase. Each phase is further discussed below.

Learning Phase. During this phase the algorithm keeps track of distances between strings. The set of observed distances during the normal period is then used to select the maximum distance that will be used for future event detection. Zero distances are dropped automatically — from experience with our experiments, most of the time distances are 0 or 0.5; 0.5 represents a 1-character difference in the string; by default adjacent characters have a distance of zero, eg. a and b have a 0 distance, but a and c have a distance of 0.5. The algorithm can be modified to drop 0.5 distances and choose not to store them as they represent minor changes.

When the learning phase completes — after a given time period elapses — the maximum distance observed is selected. The learning phase passes through $\frac{1}{4}$ of the total data in our experiments, and in the sensor-mote implementation this is not known in advance, so either the user of the system has to specify a fixed learning phase duration or otherwise the algorithm can set an arbitrary learning limit and test it against a counter incremented every time the timer fires (after some arbitrary time period has elapsed, stop the learning phase). The learning phase will largely depend on the data type monitored: for instance if we monitor light readings, then a reasonable choice for a learning phase would be twenty-four hours since in a day-night cycle we would record most of light variations and their relative distances.

Once the maximum distance is set in the learning phase it doesn't change. This is the way the algorithm works at the moment, and it is also a potential limitation. We intend to investigate ways that introduce reinforcement learning in the near future. Once we have the maximum (learned) distance we can enter

the second phase of the algorithm which is continuous (it is a function called for every new reading entering the set).

Initial Detection Phase. SAX is called to convert adjacent regions of the readings set and calculate the distance of the two strings. The readings set is essentially a sliding window of the streaming sensor data. It is stored in a data structure of a fixed size; every single new reading that enters the structure at the front, causes the oldest reading at the back to be dropped, working very much like a FIFO queue. In addition, this phase is optimised by not calculating distances for identical strings; these are dropped in silence. Generally there are two alternatives for this stage: to either call the conversion and distance comparison function at every timer tick (as shown on the algorithm) or to call it every t timer ticks. In the latter case, t has to be equal to the size of the readings set to ensure no events are undetected and a detection latency is introduced which is at worst t timer ticks. At the time of this writing we are investigating the trade-offs between the two alternatives.

Escalation Phase. The Escalation Phase is an optional, temporally deferred phase that searches through the whole readings set in order to determine if the underlying process continues to change in the future — the detection phase indicates if something has changed and the point of the change while the Escalation phase essentially determines whether the point of change was a true or false positive and whether the physical process continues to change. Usually, a defer period of 5 data points (or timer fires) is enough to distinguish a true from a false positive, but this largely depends on the sampling frequency.

During the Initial Detection Phase we pass two temporally adjacent subsections of the buffer to the conversion and comparison function — we do not pass the entire readings set (unless we use the *every t timer ticks* variant described in the previous phase). The reason we pass subsections to the function and not the entire set is because we aim to minimise the computation overhead; since we choose to call the function at every timer tick, we must ensure that the function completes before the timer expires. We also need to cater for other components such as the radio and ADC competing for CPU time.

If we choose to call the Escalation Phase, we post a `task` that is added to the queue. It takes the entire readings set and searches through it to determine whether the event was a true positive and if the process continued to change or if it stopped. The searching is done by varying the SAX window size, starting with a small value and progressively increasing it until the size of the set is reached. The compression ratio is also, optionally, reduced. For instance if the previous phase was using a $4:1$ compression, this phase can reduce compression to $2:1$ to produce more accurate results. Both of these actions (increasing the SAX window and reducing the compression) will expectedly produce higher distances. If these distances are sufficiently higher than the initial detection distance, then we can report that the process monitored either continues to change and/or that the initial detection was a true positive. Sufficiently higher is quantified by multiplying the initial detection distance by a factor empirically learned and dependent upon the process monitored and the data type.

The performance of this phase will very much depend upon the type of the event and process monitored. If an underlying process changes gradually from a normal state a to an abnormal state b, and then it remains in this abnormal state which is stable in terms of difference between data points, the escalation call will able to confirm the change, provided the change from state a to b is in the readings set. The size of the readings set is limited by the sensor's RAM and therefore if the defer period from the occurrence of the event (change of states from a to b) is large, and all the readings in the set are now stable the escalation phase will flag the report as a false positive as distance will be smaller than the initial reported distance. This indicates that the escalation phase can not be a general-purpose call but it has to be tailor-made to the underlying process monitored and the the event type.

4 Experimental Results

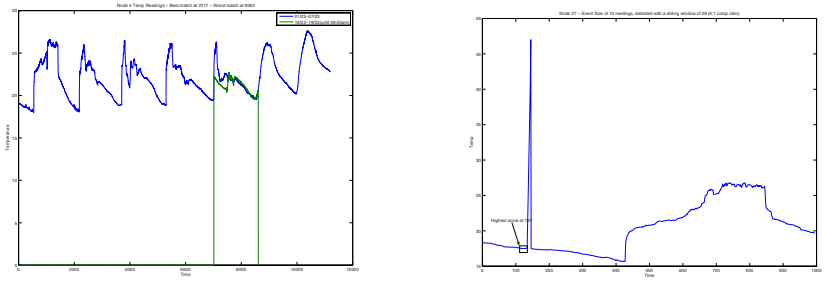
All the experiments in this section were ran in MATLAB using real sensor data that was fed to the algorithm in a streaming fashion with the use of a `Timer` object to simulate sensor network data acquisition.

4.1 Data from an Indoor Deployment

The majority of the experiments were ran on the Intel [8] data set, which is a collection of approximately 2.3 million readings from 54 sensor nodes collecting temperature, relative humidity, light and voltage. On our experiments we have used all the readings from all the nodes except voltage readings. Voltage is quite well-behaved in a sense that voltage readings do not fluctuate greatly. In contrast, light is the most unpredictable attribute and it tends to fluctuate a lot, partly due to human factors (e.g. a person switching a light on or off).

Due to space limitations, we will not discuss the outcome of every single experiment; instead we will present a few selected cases.

In the example shown on Figure 1 (a), we are searching for a specific pattern — we are looking in a week's segment of data for a pattern that is not included in that week but the most similar pattern is found. The search for a pattern works in the following manner: we take a set of data points which we convert to a string. This string represents the pattern we are searching for. Searching is accomplished by comparing this string to a string representation of the entire dataset (this can be done by using alignment algorithms instead of standard distance used in most of our trials); if distance becomes zero then it means that the exact same pattern has repeated (e.g. the event has occurred). If distance approaches zero then it means that a very similar pattern is observed. In the second example (Figure 1 - (b)) we did not supply a pattern but instead we were looking for the most interesting change. The most interesting change in this case is the spike shown on the figure — which was successfully detected. Searching for the most interesting change is the non-parametric version of the algorithm that trains on a normal set and then uses distances learnt for future detection (described in more detail in section 3.1).

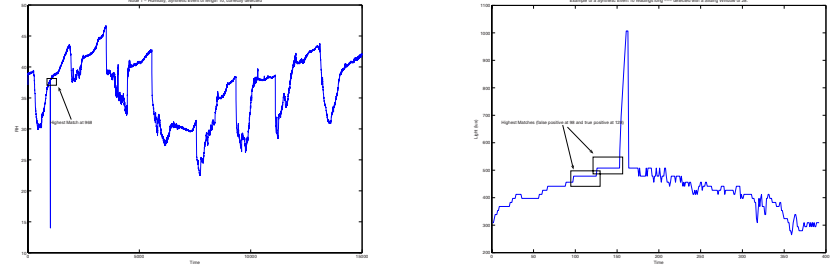


(a) Temperature Readings at node 4 (b) Temperature Readings at node 27

Fig. 1. Temperature Readings at nodes 4 and 27 — (a) approximate detection — in this case we are searching for a pattern (the readings of one day) in the readings for one week; the closest match is shown delimited by the two vertical lines and (b) non-parametric detection of a synthetic event; the spike in the graph represents an abrupt change in values — we successfully detect the exact point of change

Similarly for humidity (Figure 2 (a)), we were searching for interesting changes (in this instance, a synthetic event corresponding to the spike shown in the figure) without supplying a pattern or a threshold. One can see in the diagram that the change is correctly identified (shown by the small bounding box and arrow).

The Light attribute is somewhat different in a sense that it changes frequently and the changes are aperiodic — for example they occur whenever a human switches a light on or off. Having said that, our algorithm using SAX still performed well enough on the Light attribute. In some cases there were a few false positives, but no false negatives. This is shown in Figure 2 (b) that contains a false (first bounding box) and a true positive (second bounding box).

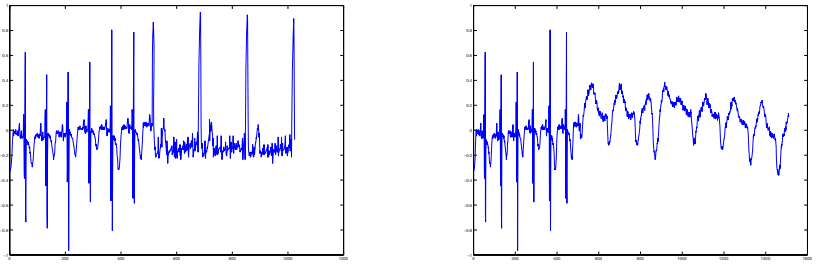


(a) Humidity Readings at node 1 (b) Light Readings at node 1

Fig. 2. Humidity and Light Readings at node 1 — (a) non-parametric detection of a synthetic event correctly identified, shown by the arrow and (b) non-parametric detection - the first match is a false positive (shown by the first bounding box) and the second match is the true positive. The false positive was discarded in the escalation phase of the algorithm, as the second match was reported with a much higher distance than the initial detection.

Overall, we were satisfied that the algorithm can successfully detect complex events of different sizes on different attributes. We had no false positive reports for Temperature and Humidity attributes, and real and synthetic events were detected accurately (accuracy in this sense refers to reporting the event very near at the beginning of its occurrence, typically within 4−8 data points). On the light attribute we had approximately 13 per cent false positives and 2 per cent false negatives. This is not entirely surprising especially if we take into account the arbitrary value changes that are largely unpredictable. Outdoor environmental monitoring data is usually more well behaved and changes can be predicted more accurately.

Finally it is worth mentioning that we have experimented with both synthetic and real events (we have used synthetic events in the parameter-free scenario, while we have used real events in the pattern-matching scenario).



(a) ECG — normal turning to Super Ventricular

(b) ECG — normal turning to Malignant Ventricular

Fig. 3. ECG — Two different ECG datasets that start normal but change at point 512. We successfully detected the point of change in both cases.

4.2 ECG Data

We have also tested our approach on ECG data obtained from [13]. We have tested two cases (Figure 3), the first one being a normal ECG that turns to Super Ventricular and the second case was normal ECG turning to Malignant Ventricular. The detection was accurate in both cases and our algorithm using SAX managed to detect the exact points of change.

5 Future Work

We have implemented a simple first version of our algorithm for the TMote Sky platform [1]. The current implementation runs on both the TinyOS2 [14] and the Contiki OS [15,16,17]. The current implementation runs solely on motes, so we are not assuming a tiered architecture but rather aiming autonomous operation. However, we aim to investigate the performance cost in comparison with

a tiered architecture where are all the motes report readings to a base-station that is not resource-constrained. The base-station will then perform the symbolic conversion and distance comparison and will make workload decisions and task the motes accordingly. One simplified execution scenario may involve tasking motes in areas of the network with high activity (e.g. reporting values that denote eventful underlying processes) to sample at higher frequencies while tasking motes in areas of the network with low activity to sample readings at low frequencies. Sleep scheduling decisions can be made based on distance information too; for example if distances over reported readings are more or less constant then a percentage of motes can be asleep or at a Low-Power Listen state — conversely when something interesting occurs (denoted by increasing distances over reported readings) explicit wake-up calls can be issued.

Our aim is to explore the trade-offs between the autonomous operation and tiered architecture. In addition we will also investigate the performance difference between the two operating systems; the TinyOS execution model is somewhat different to that of ContikiOS; as a result we expect a noticeable performance difference.

The work presented is work-in-progress and we continue to evaluate our approach with different data sets. At the time of this writing we are running experiments on motion sensor data [18] collected by the Mitsubishi Electric Research Labs (MERL) — this data is approximately 30 million raw motion readings collected by over 200 sensors during a year. We are also using hospital patient data to identify Cystic Fibrosis patients from breath samples collected using the Cyranose 320 [19] (essentially this is a classification rather than event-detection, but nevertheless we intend to evaluate the usefulness of our approach for different application settings).

Our plans for future work are further discussed in the subsections below.

Dynamic Sampling Frequency Management. As mentioned in section 3, this refers to the desired ability for individual nodes to make autonomous decisions about the Sampling Frequency, based on the distances monitored over adjacent symbolic representations of readings. A high relative distance means that the underlying physical process monitored is undergoing some change. We mention the word relative because distance is very much relative to the attribute being monitored and other application-specific factors. Please note that not every application will be able to benefit from this. For example monitoring an ECG requires more-or-less static sampling frequency and the minimum is bound by the time it takes for a single heartbeat. Other applications such as environmental monitoring can benefit greatly from adjustments in the sampling frequency. From our experience with the experiments on the Intel Lab data set, we notice that distance for great periods of time is zero or less than one (for example the light attribute during the night is almost constant — having the motes to sample at a fixed interval during nighttime is wasteful). During those times, the sampling frequency can be decreased until change starts to occur again. Once distance begins to increase, sampling frequency can be increased accordingly. And this brings us to the next point.

Local Group Coordination. So far in our discussion we have assumed that our algorithm is ran autonomously and in isolation by single nodes. And this is currently the case, but in the near future we aim to explore local coordination in such a way that the sensor field is split into regions where only two nodes are adequate to cover a specific region. The number two may seem arbitrary, but we choose two nodes so one can act as a fail-over (other choices are also valid). This can a huge energy-saving impact, especially on dense deployments. In simple terms, it means that the majority of the nodes in the network can be asleep for most of the time. The two awake nodes per region effectively monitor distances and can issue wake up calls to the rest of the nodes if they sense a big change. "Big" is defined by observing a normal period and learning what constitutes normal change (in terms of distances), as described in section 3.1. This form of sleep-scheduling can be performed in-the-network without any human interaction at all.

Reinforced Learning. We also aim to explore the possibility for reinforced learning in the machine learning sense. At present, our algorithm trains for some time and then, when it has learned an appropriate value for distance, it stops. We are planning to test our hypothesis for reinforcement learning, where learning is an ever-lasting process and new readings can affect the distance used for event detection.

Plan an Indoor Deployment. We plan to deploy a number of motes running our algorithm to monitor changes in temperature (and other attributes) in one of the university buildings.

Evaluate the Potential Use of Markov Models. We have already implemented an approach based on Markov Models in MATLAB. However, maintaining the two matrices (transition and emission) needed by such a model is a computationally-intensive process. We plan to investigate if an implementation for the sensor motes is feasible. Event-detection using probabilities is an alternative to the algorithm we currently use (which is distance-based rather than probability-based).

Our first priority is to implement and evaluate dynamic sampling frequency management and group coordination. The latter is a by-product of the former, and in our opinion the two combined can bring significant energy savings to networked sensors by making them context-aware and responsive to change.

6 Related Work

The approach by [20] is the one that come conceptually closer to our framework: in that work the authors classify gestures by comparing them to strings and performing string matching. The main difference with our approach is that we primarily aim for complex event detection and in addition we do generalise our approach for many application scenarios. In addition we offer non-parametric

event detection. In the remainder of this section we will discuss other work that is related or has affected our own research.

One of the first papers that presented an Event-Based Approach was Directed Diffusion [21]. In that approach a node would request data by sending interests which are conceptually similar to subscriptions in a publish/subscribe system. Data found to match those interests is then sent towards that node. A different framework that is based on event classification, is the Online State Tracking [22] approach. This technique consists of two phases: the first phase is the learning process where new sensor readings are classified to states and the second phase is the online status monitoring phase during which nodes are collaborating to update the overall status of the network. This is interesting work in a sense that it moves away from individual nodes' readings and views the whole network as a state machine.

Another event-based technique based on thresholding is Approximate Caching [23] whereby nodes only report readings if they satisfy a condition. A more recent paper [24] suggests a mixture of hardware and software as a solution for detecting rare and random events. The event types they consider are tracking and detecting events using the eXtreme Scale Platform (XSM) mote equipped with infrared, magnetic and acoustic sensors. Central to their architecture is the concept of passive vigilance, which is inspired from sleep states of humans where the slightest noise can wake us up when we are asleep. This is implemented with Duty Cycling and recoverable retasking. A similar approach [25] proposes a sleep-scheduling algorithm that minimises the surveillance delay (event detection delay) while it maximises energy conservation. Sleep scheduling is coordinated locally in a fair manner, so all nodes get their fair share of sleep. A minimal subset that ensures coverage of the sensing field is always awake in order to be able to capture rare events. Sleep scheduling is related to our approach since we aim to introduce Dynamic Sampling Frequency Management and base upon it a sleep coordination mechanism. This was discusses in more detail in section 5.

A first paper that addresses the need for complex event detection is the one by Girod et al [26]. In their work the authors suggest a system that would treat a sequence of samples (a signal segment) as a basic data type and would offer a language (WaveScript) to express signal processing programs as declarative queries over streams of data. The language would be able to execute both on PCs and distributed sensors. The data stream management system (called WaveScope) combines event-stream and data management operations. Unfortunately, the paper describing the system is a position paper so there is only a high-level description of the architecture (and no current implementation or mention of plans).

A paper that falls under both the Event-Based and the Query-Based subcategory, is the one describing REED [27]; REED is an improvement on TinyDB [28]. Basically it extends TinyDB with its ability to support joins between sensor data and static tables built outside the network. The tables outside the network describe events in terms of complex predicates. These external tables are joined with the sensor readings table, and tuples that satisfy the predicate indicate readings of interest (eg. where an event has occurred).

A somewhat different method that supports geographic groupping of sensor nodes was presented in Abstract Regions [29,30]. Abstract Regions is essentially a family of spatial operators for TinyOS that allows nodes to form groups with the objective of data sharing and reduction within the groups by applying aggregate operators such as `min, max, sum`, etc. The work by [31] extended the types of aggregates supported by introducing (approximate) quantiles such as the median, the consensus, the histogram and range queries. Support for Spatial Aggregation was also suggested by [32] where sensor node would be grouped and aggregates would be computed using Voronoi diagrams. Another approach [33] models the sensor network as a distributed deductive declarative database system. Their suggested method allows for composite event detection, and the declarative language used (SNLog) is a variant of Datalog.

7 Conclusions

We have presented our approach for detecting complex events in sensor networks. Complex events are sets of data points hiding interesting patterns. These patterns are difficult or even impossible to capture using traditional techniques such as thresholds. Moreover, often the users of the system may not know what they are looking for, a priori.

We use SAX to convert raw real-valued sensor readings to symbolic representations and then we use a distance metric for string comparison. Our framework can either perform exact-matching or approximate-matching of some user-supplied pattern. Moreover, it can perform non-parametric event detection; in this case the user need not specify anything. The sensor trains for some time and over some data that is known to be normal and it learns appropriate values for distance. Then, once training is finished, the learned distances are used for the detection phase.

Our implementation currently runs on the Tmote Sky platform on TinyOS2, but we have an implementation for ContikiOS for comparison purposes. Our approach appears to match very well the unique resource constraints of sensor networks.

Acknowledgements

The authors would like to thank Dr Eamonn Keogh who patiently answered our questions regarding SAX and provided relevant code and test data.

References

1. MoteIV Corporation: TMote Sky, `http://www.moteiv.com/products/tmotesky.php`
2. Katsiri, E., Ho, M., Wang, L., Lo, B.: Embedded real-time heart variability analysis. In: Proceedings of 4th International Workshop on Wearable and Implantable Body Sensor Networks (2007)

3. MIT House_n Project: http://architecture.mit.edu/house_n/
4. Biosensornet: Autonomic Biosensor Networks for Pervasive Healthcare, http://www.doc.ic.ac.uk/~mss/Biosensornet.htm
5. Dutta, R., Hines, E., Gardner, J., Boilot, P.: Bacteria classification using Cyranose 320 electronic nose. In: BioMedical Engineering Online (2002)
6. Doolin, D., Sitar, N.: Wireless sensors for wildfire monitoring. In: Proceedings of SPIE Symposium on Smart Structures and Materials (2005)
7. Firebug: Design and Construction of a Wildfire Instrumentation System Using Networked Sensors, http://firebug.sourceforge.net/
8. Madden, S.: Intel Lab Data (2004), http://berkeley.intel-research.net/labdata/
9. Hailes, S., Coulouris, G., Hopper, A., Wilson, A., Kerwin, D., Lasenby, J., Kalra, D.: SESAME: SEnsing for Sport And Managed Exercise, http://www.sesame.ucl.ac.uk/
10. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, ACM Press, New York (2003)
11. Keogh, E., Lin, J., Fu, A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In: Proceedings of IEEE International Conference on Data Mining, IEEE Computer Society Press, Los Alamitos (2005)
12. Keogh, E., Lonardi, S., Ratanamahatana, C.A.: Towards parameter-free data mining. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press, New York (2004)
13. Keogh, E.: The Time Series Data Mining Archive, http://www.cs.ucr.edu/~eamonn/TSDMA/index.html
14. TinyOS: An open source OS for the networked sensor regime, http://www.tinyos.net/tinyos-2.x/doc/html/overview.html
15. Dunkels, A., Finne, N., Eriksson, J., Voigt, T.: Run-time dynamic linking for reprogramming wireless sensor networks. In: SenSys 2006: Proceedings of the 4th international conference on Embedded networked sensor systems (2006)
16. Dunkels, A.: The Contiki OS, http://www.sics.se/contiki/
17. Dunkels, A., Schmidt, O., Voigt, T., Ali, M.: Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In: SenSys 2006: Proceedings of the 4th international conference on Embedded networked sensor systems (2006)
18. Workshop on Massive Datasets: MERL Data, http://www.merl.com/wmd/
19. Smiths Detection: Cyranose 320, http://www.smithsdetection.com/eng/1383.php
20. Stiefmeier, T., Roggen, D., Trster, G.: Gestures are strings: Efficient online gesture spotting and classification using string matching. In: Proceedings of 2nd International Conference on Body Area Networks (BodyNets) (2007)
21. Govindan, C.I.R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (2000)
22. Halkidi, M., Kalogeraki, V., Gunopulos, D., Papadopoulos, D., Zeinalipour-Yazti, D., Vlachos, M.: Efficient online state tracking using sensor networks. In: MDM 2006: Proceedings of the 7th International Conference on Mobile Data Management (2006)
23. Olston, C., Loo, B.T., Widom, J.: Adaptive precision setting for cached approximate values. In: Proceedings of SIGMOD Conference (2001)

24. Dutta, P., Grimmer, M., Arora, A., Bibyk, S., Culler, D.: Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In: IPSN 2005: Proceedings of the 4th international symposium on Information processing in sensor networks (2005)
25. Cao, Q., Abdelzaher, T., He, T., Stankovic, J.: Towards optimal sleep scheduling in sensor networks for rare-event detection. In: IPSN 2005: Proceedings of the 4th international symposium on Information processing in sensor networks (2005)
26. Girod, L., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H., Madden, S.: The Case for a Signal-Oriented Data Stream Management System. In: CIDR 2007 - 3rd Biennial Conference on Innovative Data Systems Research (2007)
27. Abadi, D.J., Madden, S., Lindner, W.: Reed: robust, efficient filtering and event detection in sensor networks. In: VLDB 2005: Proceedings of the 31st international conference on Very large data bases (2005)
28. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. 30(1) (2005)
29. Welsh, M.: Exposing resource tradeoffs in region-based communication abstractions for sensor networks. SIGCOMM Comput. Commun. Rev. 34(1) (2004)
30. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: NSDI 2004: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (2004)
31. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: new aggregation techniques for sensor networks. In: SenSys 2004: Proceedings of the 2nd international conference on Embedded networked sensor systems (2004)
32. Sharifzadeh, M., Shahabi, C.: Supporting spatial aggregation in sensor network databases. In: GIS 2004: Proceedings of the 12th annual ACM international workshop on Geographic information systems, ACM Press, New York (2004)
33. Chu, D., Tavakoli, A., Popa, L., Hellerstein, J.: Entirely declarative sensor network systems. In: VLDB 2006: Proceedings of the 32nd international conference on Very large data bases (2006)