Efficient Predictive Monitoring of Wireless Sensor Networks

Abstract: Wireless Sensor Networks (WSN) are deployed to monitor physical events such as fire, or the state of physical objects such as bridges in order to support appropriate reaction to avoid potential damages. However, many situations require immediate attention or long reaction plan. Therefore, the classical approach of just detecting the physical events may not suffice in many cases. We present a WSN level event prediction framework to forecast the physical events well in advance. Our approach is generic and allows to predict also the network events such as network partitioning, thus supporting proactive self* actions. For example, by monitoring and subsequently predicting trends on network load or sensor nodes energy levels, the WSN can proactively initiate self-reconfiguration to meet its desired operational requirements. The framework collects the state of a specified attribute on the sink for a certain duration of time. For this we propose an efficient spatio-temporal compression technique. Based on the collected history of attribute values, the future state of the targeted attributes are predicted using time series modeling. The forecasted future state is then used to detect generalized future events through our proposed event detection algorithm. Additionally, the proposed framework is adaptable to cover multiple application domains. Using simulations we show our framework's enhanced ability to accurately predict the network partitioning, accommodating parameters such as shape and location of the partition with very high accuracy and efficiency.

Keywords: Wireless Sensor Networks, Predictive Monitoring, Time Series Analysis, Spatio-Temporal Compression, Event Detection and Prediction.

1 Introduction

Wireless Sensor Networks (WSN) typically entail an aggregation of both sensing and communicating sensor nodes to result in an ad hoc network linking them to the base station or sink. The sensor nodes typically possess limited storage and computational capabilities and require low-energy operations to provide longevity of operational time.

WSN's are deployed for monitoring different environmental attributes. Based on the sensed data ranges, the corresponding reaction decisions are carried out. The decisions to take actions are triggered by some events happening in the network. For example, while monitoring pressure in a certain facility an event could be triggered to indicate either high or low pressure. Similarly, we can have many events related to numerous attributes. In addition to the environmental events, there are also the network events to be considered such as network partitioning. Various works exist for detecting different discrete events (Yick et al., 2008) such as, fire detection (Yu et al., 2005) and network partitioning (Shrivastava et al.,

2005; Rost and Balakrishnan, 2006; Shih et al., 2007). Most of these efforts develop excellent foundations, however, are tailored for specific scenarios. Other works do consider generic scenarios (Xue et al., 2006), but they suppose the event to take specific shapes and patterns. Also, all of these efforts focus on detecting the events *after* the events have already occurred. It could be already too late to react to many events if the traditional approach of detecting and reacting to the events is followed. Consequently, it is either insufficient or inefficient just to react to the events. For example, if we detect network partitioning, the repair might require a long time and the required resources may not be available. Meanwhile, the functionality of network and hence the monitoring, which is the main objective of deployment, will be lost. Thus, reporting of such events beyond simple monitoring becomes highly useful if these events can be predicted in advance. The ideological shift from detecting the events to predicting them provides enough time window to take appropriate autonomic actions. Consequently, we could avoid or delay events from happening. Multiple efforts also exist for predictions (Mini et al., 2002; Tulone and Madden, 2006; Landsiedel et al., 2005; Wang et al., 2007). However, most of them are either limited to predicting specifically a certain attribute such as energy or provide only node level short-term prediction for data compression to minimize data to be reported from the network.

It is very useful to combine generic prediction techniques with generalized event detection to predict the events and to carry out self* actions well in advance. To the best of our knowledge there exists no work that proposes generic event prediction. In this paper, we develop a generic framework to predict the events. The framework predicts the future states of the network for the attribute of interest, e.g., temperature or residual energy. The developed generic event detection technique is used on "predicted future state of attributes" to effectively forecast the events. We target long-term predictions that require the history of the attribute to be long enough to contain all system dynamics. However, a sensor node does not possess the computational resources required to model the complex dynamics in attribute values to accurately predict the future states. Hence, we collect multiple profiles of the considered attribute and conduct modeling on the sink. We refer to the collection process of attribute values from network as *profiling*. Accordingly, a profile is the state of the attribute in the network at a specified instance of time. We also propose data collection techniques to efficiently and accurately collect such a history of the attribute from the network. On this background this paper makes four specific contributions, namely

- Generalized framework design for sink-aided attribute profile prediction, allowing to predict varied physical and network events.
- Efficient data collection technique based on spatio-temporal compression, minimizing both spatial and temporal data redundancies while collecting attribute profiles.
- Generic event detection technique to detect events from the predicted profiles.
- Case study of network partition as validation for our efficient predictive monitoring framework.

The paper is organized as follows. Section 2 discusses the related work. Section 3 details the key preliminaries for the framework. Section 4 details our approach for predictive WSN monitoring. The case study is presented in Section 5 and its simulation evaluations in Section 6. Section 7 presents summary conclusions and outlines future directions.

2 Related Work

In WSN literature a variety of work addresses event detection (Yick et al., 2008). The most relevant one to our event detection strategy is (Xue et al., 2006), where the authors investigate map based event detection. The approach requires the user to (a) specify the distribution of an attribute over space and (b) the variation of distribution over time incurred by the event. Three common types of events are defined namely pyramid, fault and island. In contrast, our detection technique is independent of event shape thanks to our generic regioning algorithm. Furthermore, we apply the detection technique on predicted profiles allowing to predict events rather than just detecting them. (Banerjee et al., 2008), presents a technique to detect multiple events simultaneously. They employ a polynomial based scheme to detect event regions with boundaries and propose a data aggregation scheme to perform function approximation of events using multivariate polynomial regression. Our work in addition to the capability of detecting multiple events, can predict events beforehand. Various other works exist that address specific event scenarios such as partition detection (Shrivastava et al., 2005), and fire detection (Yu et al., 2005). These specific solutions do not feature portability to adapt to different application scenarios.

There is a variety of work for monitoring WSN's and prediction of certain attribute. (Landsiedel et al., 2005) predicts the power consumption in WSN. (Mini et al., 2002) proposes a network state model to predict the energy consumption rate and constructs energy map accordingly. In (Wang et al., 2007), authors focus on predicting multimedia networks energy efficiency. These works concentrate specifically on energy, also they do not provide any extension to predict other attributes. Authors in (Mamei and Nagpal, 2007) propose inference mechanism using Bayesian network to detect anomalies. We provide a generic framework to predict variety of events that might happen in future.

As we present a case study for network partition prediction, we discuss the related work in this respect. In (Shrivastava et al., 2005), partition detection has been addressed for a sub-class of linearly separable partitions, i.e., cuts. Memento (Rost and Balakrishnan, 2006) continuously collects connectivity information at the sink to be able to detect network partitioning. The partition avoidance lazy movement protocol for mobile sensor networks (Shih et al., 2007) is a decentralized approach, where a node periodically collects the position of all its neighbors and checks if at leat one neighbor is located in a small angle towards the sink. If no neighbor is located in this "promising zone", the node suspects network partitioning and moves to avoid it. Based on our event prediction framework as an example we propose a solution that is generalized and not dependent on the shape, size or location of the partition. Moreover, our framework provides for prediction of network partitioning rather than just the detection.

3 Preliminaries

We now describe the system model and the requirements driving our approach.

3.1 System Model

We consider a WSN composed of N static sensor nodes and one static sink. Sensor nodes are battery powered and usually entail limited processing and storage capabilities. Sensor nodes are assumed to know their geographic position either

using distributed localization methods (He et al., 2005) or GPS. A typical WSN deployment may contain hundreds or thousands of sensor nodes with varying densities according to the coverage requirements. We consider an arbitrary node distribution, provided the network is connected at deployment time. We assume all sensor nodes to be homogeneous. Hence, the sensor nodes have the same transmission range R and same initial battery capacity. We consider that nodes crash due to energy depletion only. We assume a reliable data transport protocol, e.g., (Shaikh et al., 2010) to transports the data from sensor node to the sink. We assume the events for predictions to be happening over a longer period of time, e.g., events that may take hours, days or even months to develop. We consider that events are (a) not spontaneous, (b) spatially correlated, (c) do not depend discretely on a single node, and (d) display attribute trends that can be predicted.

3.2 Requirements on the Framework

We identify the following requirements on the framework. First, it should be *lightweight*, i.e., its creation, management and usage requires minimal resources with respect to energy. Second, we desire the framework to *long-term* predict attribute profiles, hence the events *accurately*. Depending on the context of the problem, long-term may mean hours, days or even months that should be enough to activate a self* mechanism to support autonomic actions. Third, we desire the framework to be *generic* to adapt to prediction of varied event types.

4 Efficient Predictive Monitoring

We develop the proposed framework in a modular manner for it to be generically applicable to a variety of scenarios. The framework consists of three phases, i.e., *data collection phase*, *prediction phase* and *event detection phase*. In the data collection phase, attribute values related to an event are periodically but efficiently fetched from the network on the sink. The prediction phase is used for predicting future states of the network for the interested attribute using the previous history of attribute fetched from the network during data collection phase. The main objective in event detection phase is to detect events in the predicted state of the network, obtained in prediction phase, essentially predicting the events. The techniques we propose in each phase are independent of the attribute to be monitored, thus fulfilling generality requirements of our framework. It is important to highlight that we do not limit the framework to only these techniques. Rather, for a particular implementation, specialized additional techniques can be easily accommodated due to the framework's modular structure. These phases are individually detailed in the following sections.

4.1 Data Collection Phase

From the nature of the problem we can expect an efflux of data (sensed attribute values) from the network towards the sink. A simplistic periodic approach to collect data from each sensor node on the sink would lead to high communication and energy overhead on sensor nodes rendering the whole framework impracticable. Our framework addresses this issue by exploiting the inherent redundancies to reduce the amount of data to be transported to the sink without sacrificing the accuracy of the collected data.

4.1.1 Data Redundancy in WSN

We first describe the redundancies present in WSNs, and how these redundancies can be exploited in general. Subsequently, we present our approach that efficiently utilizes both spatial and temporal redundancies for compression.

Spatial Redundancy - Reduction Through Clustering: By its basic nature, a WSN involves redundancy in node deployment that yields redundant sampling of the environment. Furthermore, spatial distribution of the attributes in the environment such as temperature, pressure, etc., tend to be similar over large contiguous area than just in the neighborhood of redundant nodes. To avoid spatial data over-sampling, not all the nodes need to send their samples (spatial compression). The existing approaches take into account spatial redundancy by forming clusters of similar valued nodes (Gedik et al., 2007). Authors in (Gedik et al., 2007) further break the clusters into sub-clusters to select fewer sampling nodes. However, they have to re-execute cluster construction algorithm repeatedly to maintain these bigger clusters. There are also other variations like (Solis and Obraczka, 2005) that form clusters and send updates to sink if there is any change on the edge of the cluster.

Definition 1 *A* time series *is a sequence of data points x_t considered as a sample of random variable $X(t)$, typically measured at successive times. The time series can be modeled to predict future values based on past data points.*

Definition 2 *A* stationary random process *exhibits similar statistics in time, characterized as constant probability distribution in time. However, it suffices to consider the first two moments of the random process defined as weak stationary or wide sense stationary (WSS) as follows:*
1. The expected value of the process ($E[X(t)]$) does not depend on time. If $m_x(t)$ is the mean of $X(t)$ then
$E[X(t)] = m_x(t) = m_x(t+\tau)\ \forall\tau \in \Re$
2. The autocovariance function for any lag τ is only a function of τ not time t
$E[X(t_1)X(t_2)] = R_x(t_1, t_2) = R_x(\tau, 0)\ \forall\tau \in \Re$

Definition 3 *$X(t)$ is an Autoregressive Moving Average Process $ARMA(p, q)$ process of order (p, q) $p, q \epsilon \aleph$, if $X(t)$ is WSS and $\forall$ t,*

$$X(t) = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q} \quad (1)$$

where Z_t is white noise with mean zero and variance σ^2, denoted as $WN(0, \sigma^2)$.

Temporal Redundancy - Reduction Through Piecewise Modeling: We primarily observe for a given node the monitored attribute values are often dynamic in nature, however, they are usually statistically correlated in time. This correlation in time can be exploited by abstracting the attribute values as a time series (Def. 1). Consequently, we can compress the data by modeling the raw data samples and send only the model parameters to the sink (temporal compression).

The generic time series modeling is not possible on a node because of its limited storage and computational capabilities. Therefore, a time series on a node is better modeled piecewise. This also avoids having to model complex time series dynamics. Consequently, the models are still simple enough to be evaluated on sensor nodes. As in (Tulone and Madden, 2006), nodes maintain short history of data samples (sensed attribute values). A $3rd$ order autoregressive model (AR3) is fit to the data, which is only a particular case of the ARMA(p, q) model

(Def. 3), for $p = 3$ and $q = 0$. AR3 gives a good compromise between complexity and predictability (hence compressibility). Nodes estimate next values according to this model. If the model is no longer valid for the new data a new model is constructed. A few values that do not fit the model are termed as *outliners*. The track of outliners to the model is kept explicitly. A minor optimization over this basic scheme is to group nodes in 1-hop cluster (C_i). Using this optimization the nodes that are in each other's transmission range build only one model instead of each one building its own model. Consequently, only cluster heads (Hc_i) maintain, update and send the models to the sink. A node joins a cluster C_i if its attribute value is within the allowed maximum threshold (ΔMC_i) between the Hc_i and the cluster members (Mc_i). Two clusters may overlap, i.e., the members of one cluster may communicate with the Hc_i of the other neighboring cluster. Hc_i periodically broadcasts its attribute value to its members Mc_i so they ensure that they are also within the error bounds ΔMc_i of the value of Hc_i. Otherwise, they leave this cluster and join another cluster for which error is within ΔMc_i or build a new cluster if they do not fall within ΔMc_i of any of the surrounding Hc_i.

4.1.2 Optimized Spatio-Temporal Compression Scheme

Our main design objective is to reduce the overhead while maintaining high accuracy of data. We propose a hybrid compression scheme to reduce the reported data by exploiting both spatial and temporal redundancies.

The spatial and temporal compressions are complementary schemes. The current literature optimizes for one scheme (Solis and Obraczka, 2005) or the other (Tulone and Madden, 2006). Applying both schemes together is challenging. Clustering algorithms group nodes based on the similarity of the values. However, value based spatial clustering can not guarantee that such achieved grouping will hold for long time. Moreover, the temporal compression is based on the assumption that the entity to be modeled demonstrates a certain behavior in statistical terms such that it fits a certain model for certain duration of time. Hence, a large cluster may not live long enough to be justified to be modeled and breaks up into further clusters. Therefore, the temporal compression schemes are very efficient on node level or in a very small neighborhood but suffer the scalability issues when applied to model behavior of multiple nodes or clusters. Thus, we conclude that the larger the cluster, the shorter the spatio-temporal correlation holds.

We propose a very efficient approach in terms of message and energy overhead. Our approach exploits both spatial and temporal redundancies. The proposed scheme starts with 1-hop clusters 'C_i' based modeling (Section 4.1.1).The main advantage of 1-hop clusters is that it is the unit of clustering, i.e., all the higher level clusters can be formed from these clusters by merging them. They are easy to construct and maintain, i.e., through simple 1-hp beacons (Tulone and Madden, 2006). Moreover, if the model does not fit the new attribute values, C_i does not break up like larger clusters, instead a new model is built to fit to the new values. However, if the attribute values are so discretely divergent that each node has different values within 1-hop then it implies that there is no redundancy, which is contradictory to inherent redundancy in WSN.

We next describe our algorithm to determine spatio-temporally correlated regions in WSN. Subsequently, an algorithm is presented to optimally aggregate the region information to be sent to the sink.

Distributed Regioning Algorithm (DRA): Data compression for 1-hop cluster based modeling is efficient for temporal compression but is limited to 1-hop cluster area. However, as discussed before, the attributes tend to be similar in larger area than just 1-hop distance. To also exploit the spatial redundancy for C_i models, a Hc_i sends its model to its neighboring clusters. The neighbors compare the received model with their local model. If the error is under given error bounds, neighbors accept their behavior to be represented by the same model as Hc_i and forward this model to their respective neighbor clusters. This scheme generates a region that is spatially correlated for the cluster head models. Consequently, we achieve spatio-temporal compression. Now, we elaborate DRA further in detail.

Algorithm 1 DRA: Distributed Regioning Algorithm (On Sensor Nodes)

```
 1: VAR: PJN=Positive Join Notification, NJN=Negative Join Notification, Ri=Region ID, list=list of clusters, NBN=Not border Notification
 2: function growRegion()
 3: me.list.addToList(me);
 4: msg=["join request" model Ri];
 5: for C_i ∧ ∀C_i ∈ Nc_i do
 6:    if sendMsg(msg, C_i)="PJN" then
 7:       me.Nc_i.C_i=Ri;
 8:    else
 9:       me.Nc_i.C_i= -Ri;
10:       me.Nc_i.C_i.border=false;
11:    end if
12: end for
13: if C_i.Ri=me.Ri∧∀C_i ∈ Nc_i then
14:    me.border=flase;
15:    msg=["NBN" list];
16:    for C_i ∧ ∀C_i ∈ Nc_i do
17:       sendMsg(msg, C_i);
18:    end for
19:    if me.BTT then
20:       me.BTT=False;
21:       msg=["BTT"];
22:       sendMsg(msg, C_i ∧ ∀C_i ∈ Nc_i & minDist(Nc_i));
23:    end if
24: end if
25: if C_i ∧ ∀C_i ∈ Nc_i & C_i.Ri! = "" then
26:    me.transition=false;
27:    me.stable=true;
28: end if
29: function recieveMsg(msg)
30: me.Nc_i.C_i=msg.Ri;
31: if msg.type="join request" & me.Ri="" then
32:    nbrValues=expandModel(msg.model);
33:    outs=cntOuts(nbrValues,me.Values);
34:    if outs < ΔNc_i then
35:       me.regionId=msg.Ri;
36:       me.border=true;
37:       me.transition=true;
38:       scheduleMsg(call growregion());
39:       return(PJN);
40:    else
41:       return(NJN);
42:    end if
43: else if msg.type="NBN" & me.border then
44:    me.Nc_i.border=false;
45:    me.list=msg.list;
46: else if msg.type="BTT" then
47:    me.BTT=True;
48: end if
```

During the cluster formation each Hc_i exchanges its distance from the sink with its neighbor clusters (Nc_i). Thus, each Hc_i knows about its neighboring clusters and their distances from the sink. Each Hc_i starts sampling the attribute value. The local model is generated after collecting enough samples. The Hc_i waits for maximum time T_{max}, inversely proportional to its distance from the sink (C_i.distance), before initiating regioning. If wait time is denoted by T_w, then it can be given by $T_w = T_{max}/C_i.distance$. T_w biases the clusters farther from the sink to initiate the regioning earlier. The cluster that initiates the regioning has a special privilege that we refer as border traversal token (BTT). We shall elaborate the use of BTT token and biasing of DRA initiation away from sink afterwards. As given in Alg. 1, DRA is a three step procedure:

- Once a cluster head, denoted as the sender cluster C_s, has fit a model to the data after wait time T_w, it then sends its model to the heads of the neighboring clusters Nc_s (Alg. 1:L 4-12).
- A receiving cluster heads (C_r) finds dissimilarity between the received model and its own model based on a dissimilarity metric to evaluate the error between

two models. This metric can be a statistical function, a signal processing metric such as cross-correlation or time series analysis technique that we propose here. We use time series based technique because it is very efficient and that the models to be compared are derived using time series analysis. Hence, we can guarantee the accuracy level with the dissimilar error bounds parameters that we otherwise use for in-cluster dissimilarity measures. We introduce a maximum allowed error threshold (ΔNc_i) between C_s and C_r for merging. We define ΔNc_i to be the maximum number of allowed outliners to the received model. To compute dissimilarity the C_r evaluates the received model (Alg. 1:L 32). It counts the outliners by comparing C_s values (nbrValues), received as C_s model, to its sampled values (Alg. 1:L 33). If outliners are within ΔNc_i, C_r acknowledges C_s by sending a "Positive Join Notification" (PJN) to merge into the region that is represented by C_s model. Otherwise, C_r sends a Negative Join Notification" (NJN) (Alg. 1 :L 34-42). The cluster issuing a PJN considers itself on the boundary of the region (Alg. 1:L 36) and schedules itself to further propagate the model to its neighbors (Alg. 1:L 37). Consequently, every cluster head participating in the region knows its status either as the region interior or the region boundary to keep track of region boundary.

- If all neighboring clusters of C_s belong to the same region, this implies that C_s is not on the region boundary (Alg. 1:L 13-14). C_s notifies Nc_s it is not on border anymore through a "Not Border Notification" (NBN) message and transfers the aggregated clusters list of this part of the region to Nc_s (Alg. 1:L 15-18). However, this information is of interest to only the clusters on the boundary. Therefore, the list is updated by only the border clusters (Alg. 1:L 43-46). More than one border clusters receive the list, consequently multiple border clusters may have duplicated information for region interior in the end, which is then removed in the border traversal algorithm (BTA), described in Alg. 2. The cluster leaving the border status hands over the BTT token to the cluster head closest to the sink on the border if it possesses one (Alg. 1:L 19-23).

The three steps repeat until the outliners count is below ΔNc_i for C_s and C_r. This process creates a region that is spatially and temporally correlated for an attribute for the modeled time duration. Hence, using DRA we have just one model to be reported to the sink that represents the behavior of the correlated region. Fig. 1 illustrates an execution example of DRA. For illustrative purposes and simplicity we removed the duplicated arrows and show the clusters to be non-overlapping.

The DRA may be initiated by multiple clusters simultaneously. Consequently, multiple regions may grow and merge. However, no special consideration is needed for this situation, as the condition still holds that if all the clusters around a cluster are part of the same region then this cluster leaves the border cluster status. Hence, the previous border clusters within the true border of the new bigger region will automatically annihilate and the border clusters on the true border will still persist. However, multiple merging regions will generate multiple clusters having BTT token. We refer to this as multi-BTT problem and deal with it in BTA.

Border Traversal Algorithm: The information of region constituting clusters and the single model representing the region needs to be sent to the sink. We aggregate this information on one cluster head that reports to sink the aggregated information, instead of each border cluster reporting to the sink. We propose the

Border Traversal Algorithm (BTA) to aggregate the complete region information (Alg. 2). A specific probability of selection of clusters on the boundary could be set to initiate BTA. However, it might lead to selecting either no or multiple clusters to initiate border traversal. Only one cluster should send the model and the region information to the sink in order to optimally reduce the data to be transported to the sink. This cluster should preferably be the one closest to the sink to further decrease the message overhead. This is where BTT comes into play. Only the cluster possessing a BTT can initiate BTA. The DRA algorithm is biased to initiate away from the sink, so there are higher chances (though not guaranteed) that it will expand in the direction of the sink. Moreover, the BTT token is transferred to the cluster closest to the sink in DRA. Though, again it does not guarantee but biases to select cluster closest to the sink to initiate BTA.

Once the region stops growing around a cluster having BTT token, it initiates the boundary aggregation. The initiator cluster includes its distance from the sink in the aggregation message. It helps to eliminate the multi-BTT problem. The cluster holding the token sends the message to aggregate the region information. The receiving cluster updates the list by including itself and the list that it holds, collected during DRA (Alg. 2:L 13-14). However, this list may contain duplicated cluster information, therefore a cluster receiving the aggregation message runs a filter to remove the duplicated cluster information (Alg. 2:L 15). If the region has multi-BTT problem then multiple clusters initiate traversal. However, this is not difficult to handle as the traversal request initiator closest to the sink will suppress traversal request initiated by the other clusters (Alg. 2:L 5-7). It is possible that the region stops growing around the token holding cluster, it enters into stable phase and initiates BTA. However, in some other part the region may still be in transition phase and growing. After the aggregation process is triggered and reaches a cluster in transition phase, this cluster pauses the aggregation process. This cluster state information is maintained during DRA (Alg. 1:L 25-28, 37). As soon as a cluster in transition phase enters into the stable phase, it continues the aggregation process with the next border cluster (Alg. 2:L 8-12). The pause and resume mechanisms ensure that we collect the accurate border information.

Algorithm 2 BTA: Border Traversal Algorithm (On Sensor Nodes)

```
1: VAR: aggMsg=[initId=Iditiator cluster id, initDist= initiator distance from sink, list=aggregator list of border clusters], C_s=Sender Cluster
2: Function recievedMsg(aggMsg)
3: if me.Id = aggMsg.initId then
4:     sendToSink(aggMsg);
5: else if me.BTT & me.distance < aggMsg.initDist then
6:     suppress(aggMsg);
7:     return;
8: else if me.transition = true then
9:     while me.transition do
10:        pause;
11:    end while
12: end if
13: aggMsg.list.addToList(me);
14: agg.list.addToList(me.list);
15: aggMsg.list=filter(aggMsg.list);
16: sendMsg(aggMsg, C_i ∧ ∀C_i ∈ Nc_i & C_i.border!=false & C_i!=C_s);
```

DRA and BTA very effectively reduce the amount of data to be transported to the sink by transporting only a few models by finding a small number of model correlated regions for the whole network.

4.2 The Prediction Phase

The prediction phase takes place on the sink. We regenerate the profiles for the desired attribute using the collected models of regions. Generic time series

modeling techniques are then used to model the complete history of each node to predict the future profiles.

The models received on the sink in data collection phase for each region are used to regenerate the attribute history through reverse transformation. It is rather a simple procedure as sink has the models for each region, so reverse transformation comprises of solving the region model equation for each node in each cluster constituting each region. Hence, the reverse transformation generates many complete profiles of the WSN. The reverse transformation forms a temporal stack of such profiles as shown in Fig. 2. The regenerated history of each node contains all the complex dynamics. On the sink we can now take the complete history of each node and model its complete behavior using time series analysis for predictions as opposed to on node piecewise modeling. Individual models of each node can then be used to predict future values by fitting a prediction model, effectively predicting future profiles. The time series can be modeled in different ways (Wang et al., 2007). In this paper, we use the widely used time domain modeling because of its general applicability.

4.2.1 Modeling Time Series:

A time series $X(t)$ can be modeled as a process containing following components

$$X(t) = T_t + S_t + R_t \tag{2}$$

where T_t is a trend, S_t is a function of the seasonal component with known period, and R_t is the random noise component. To keep the notion of generality valid for the framework we use a well known generalized technique termed Box-Jenkins Model to model a time series containing any of these components.

4.2.2 Box-Jenkins Model:

Box-Jenkins (BJ) model predicts a time series by fitting it an Autoregressive Integrated Moving Average (ARIMA) process. The term integrated here means differencing the series to achieve stationarity (Def. 2). To fit an ARIMA process the model and the order of the model needs to be specified. The BJ model provides a guideline to select the appropriate model, i.e., either Autoregressive (AR, Eq. (3.1)) or Moving Average (MA, Eq. (3.2))

$$X(t) = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} \ \{3.1\}, \ X(t) = \theta_1 Z_{t-1} + \cdots + \theta_p Z_{t-q} \ \{3.2\} \tag{3}$$

or combination of both, i.e., ARMA process as given in Eq. (1). It also gives the guideline for the model order selection. BJ modeling is a four steps procedure:

i.) Data Preparation: BJ model requires a time series to be stationary (Def. 2). Therefore, if it contains trends and seasonal components then these should be appropriately removed. This can be achieved by either Least Square Polynomial Fitting (LSPF) or differencing as $X(t) = X(t) - X(t+u)$. For a simple linear trend, u is 1. For higher order trends or seasonal component of period s, u equals s. This operation is repeated until stationarity is achieved.

ii.) Model Identification: At this stage run-sequence plot or Autocorrelation Function (ACF) can be used to identify the stationarity of the time series and the order of the AR model. ACF for k lag is given by

$$\rho_k = \frac{\sum_{i=1}^{N-k} \left(X_i - \bar{X}\right) \left(X_{i+k} : \bar{X}\right)}{\sum_{N}^{i=1} \left(X_i - \bar{X}\right)^2} \tag{4}$$

where $\bar{X}$ is the mean value. Non-stationarity is often indicated by an ACF plot with very slow decay. Order of the AR and MA models are determined with the help of ACF and Partial Autocorrelation Function (PACF) (Montgomery et al., 2008). To automate the model selection process either Akaike's Information Criterion (AIC) or Akaika's Final Prediction Error (FPE) (Ljung, 1998) can be used. Various models can be computed and compared by calculating either AIC or FPE. The least value of AIC or FPE ensures the best fit model.

iii.) Parameter Estimation: In this step the values of the ARMA model coefficients that give the best estimate of the series are determined. Iterative techniques are used for model parameter estimation (Ljung, 1998).

iv.) Prediction: Once the modeling is complete, it is simple to predict the series values using the estimated model. It comprises of calculating the future values at next time instances and reversing all the transformations applied to the series in phase 1 for data preparation.

4.3 The Event Detection Phase

We now develop a generic event detection technique. Subsequently, using this technique we detect events in the predicted profiles of the attribute obtained in prediction phase, effectively predicting upcoming events in the network.

The main objective of our framework is to predict events. In the system model (Section 3.1) we have described the domain of the events that we are targeting in our work. These can be exemplified by detection of temperature above a certain threshold in a certain part of network that is indicative of fire. Similarly, there are many other events associated with each physical attribute such as pressure, humidity etc. Our challenge here is to design an event detection mechanism that is generic and can be ported to wide range of scenarios. To cope with this problem we use here an abstraction of maps for WSN (Def. 4). For a WSN an eMap is an energy map that represents the current residual energy of the network (Zhao, 2002), or tMap for temperature etc. Once a WSN is converted to a map for a certain attribute such as temperature, pressure etc., the events appear as regions in these maps. For example, in a tMap of WSN the part of the network that is beyond the given threshold of temperature will appear as a region in a tMap. Consequently, in our framework we define an event as a region of map whose values fall in the range of attribute values for which the event is defined. Using the abstraction of maps for WSN and regions for events we are able to keep the framework generic enough to be portable to different scenarios. Thus, the quantification of WSN space and the conversion of a WSN to a map abstraction is the key to detect generic events.

Definition 4 *To quantify the continues space of WSN profile and construct the map a grid is virtually placed over the WSN profile and each grid cell represents the aggregated attribute of all the nodes located within the grid cell. We define the resultant quantification as* Grid Map *or simply* Map.

4.3.1 Map Abstraction

In order to reach an acceptable spatial resolution with higher level abstraction of network as a map, we considered virtual grids and Voronoi diagram (Aurenhammer, 1991) techniques to segment WSN profile. Voronoi-based segmentation depends only on sensor node distribution and is static for

a given node distribution. However, we require a segmentation strategy that allows variable spatial sampling to accommodate both the physical and network parameters. Such variability also allows us to detect events from a single node to a region of the network. Grid allows such flexibility therefore, we base our map construction on grid. The virtual grid or simply *grid* divides the WSN profile into fixed size squares or *grid cells* as shown in Fig. 2. Thus, nodes that fall within a cell are grouped. For the grid map construction, two parameters must be specified. The first parameter is the grid cell size γ, which is a spatial sampling or resolution parameter. The second parameter is the aggregation value ξ that a grid cell represents. Both parameters are essential for event detection. γ defines the geographic area covered by the grid cell. The number of nodes being grouped in a grid cell is dependant on γ. It can also be seen as a zooming parameter. Hence, it can be used to decide at which level the user intends to detect the event, i.e., very detailed (zoomed-in) level of node or an overview at the level of regions. The grid cell value ξ is an aggregate of the attribute values of the set of nodes in a cell. The choice of the exact function depends on the application. For example, for temperature or pressure, it is most appropriate to average the values of the nodes in the grid cell. If ξ_{ij} is the grid cell value in the $(i,j)^{th}$ grid cell g_{ij} and v_n represents attribute value of node n in g_{ij} then ξ_{ij} is an aggregation function such as average, min, max of v_n:

$$\xi_{ij} = f(v_n) \ \forall \ n \in g_{ij} \tag{5}$$

We do not impose assumptions on the selection of γ and $f(.)$, highlighting the generality of our framework (requirement on our framework). An illustration for the selection of both parameters is given in the case study in Section 5.

4.3.2 Centralized Regioning Algorithm

As the events appears as regions in a map, we propose here a centralized regioning algorithm (CRA) that can detect the regions and their borders in WSN map, which leads to generic event detection. CRA is conceptually the same as its distributed counterpart DRA. However, it has been used here for entirely a different purpose of detecting events. The parallel between the two applications of conceptually same algorithm is that in DRA the *models* build correlated regions based on similarity of models and in CRA the *events* build regions based on similar values of attribute for an event.

The regions are formed because the attribute values fall into a certain class of values. For example, we normally classify the temperature as freezing, low, normal, high or very high. These classes also contain event class (range of values belonging to event, e.g., temperature above 500^oC for fire). This gives us more acceptable abstraction than the exact values themselves. Therefore, thresholding of values into classes becomes logical representation for event detection. Thus, to detect these events we define the *class maps* that thresholds the exact values of the cells in grid map with their class denominations. If we define class map values K as k_1, $k_2, \cdots$ for the range of the values of grid cell g_{ij} between $(\xi_2, \xi_1]$ and $(\xi_3, \xi_2] \cdots$ respectively, then a class map value is defined by

$$K = \begin{cases} k_1 \text{ if } \xi_2 < \xi_{ij} \leq \xi_1 \\ k_2 \text{ if } \xi_3 < \xi_{ij} \leq \xi_2 \\ \cdots \end{cases} \tag{6}$$

CRA (Alg. 3) takes the grid map as input and determines border and regions belonging to different classes and hence events. We refer to the resultant output as the *regions map*. CRA essentially needs a class map to group all the same class cells and determine the boundary. The process of converting to class map and determining the regions boundary are both carried out concurrently. In order to merge the cells into regions, we define attribute classes as in Eq. (6). Neighboring cells are merged to form the same region if they belong to the same class. The definition of attribute classes and fusion of same class grid cells makes CRA independent of the shape that a region takes or the number of regions (hence the number of events) in the map.

Algorithm 3 CRA: Centralized Regioning Algorithm (On the Sink)

```
 1: Var: rB= regionBorder, mB= mapBorders,
    nRB= newRegionBorder, rM= regionsMap,
    rId= regionId, nL= neighborList, Gc_xy=
    Grid cell at (x,y)
 2: rM[][]=-1;
 3: mB[][];
 4: while rM(i,j) = −1 do
 5:    rB[]=(Gc_ij ∧ rM(i,j) = −1))
 6:    dilateRegion(map, rB[], rM[], rId)
 7:    mB[rId][]=rB;
 8:    rId++;
 9: end while
10: Function dilateRegion(map, rB[], rM[], rId)
11: repeat
12:    changeInBorder=0;nRB[]=0;
13:    for Gcij ∧ ∀Gc_ij ∈ rB) do
14:       nL[] = 8Nbrs(Gc_ij);
15:       for Gc_kl ∧ ∀Gc_kl ∈ nL[] do
16:          if (class(Gc_ij) = class(Gc_kl) &
             rM(k,l) = −1 then
17:             rM(k,l)=rId;
18:             nRB.add(Gc_kl);
19:             changeInBorder=1;
20:          end if
21:       end for
22:       if class(Gc_ij) ≠ class(Gc_kl) ∧
          ∀Gc_kl ∈ nL[]) then
23:          nRB.add(Gc_ij);
24:       end if
25:    end for
26:    rB[]=nRB[];
27: until changeInBorder
```

The algorithm starts by defining all the cells as *not assigned to a region* by initializing the variable rM[]=-1. Next it searches a grid cell that has not been assigned a class yet and lists it as the border of the region, as the region itself and region border at this moment consists of a single cell (Alg. 3:L 5), cell with -1 in rM is not assigned a region yet). CRA then starts expanding/*dilating* the region (Alg. 3:L 6). To expand the region, the neighboring eight cells around this region cell are checked if they already belong to any class (Alg. 3:L 14-16), if not then they are also classified according to Eq. (6). If they belong to the same region they are assigned the same region ID and the new qualifying cells are listed as the region border, otherwise the previous cells retain their status as region border (Alg. 3:L 17-18). To further expand the region neighboring cells of each cell in the border cells are searched iteratively until no change occurs in the border of the region (Alg. 3:L 11,27), which implies the completion of the construction of a single region with its boundary. The whole process repeats again by searching a new cell that has not been assigned a region yet. It keeps on repeating until all the cells in the map are classified into their corresponding regions (Alg. 3:L 4,9).

We maintain the generality of the framework by devising a technique that does not assume any shape, size or number of events occurring in the WSN.

5 Case Study: Network Partition Prediction

We should formulate the problem according to the abstractions (maps, classes, etc.) of the framework, to use our framework for network partition prediction.

5.1 Problem Formulation

Partition detection is a complex problem as physical and network parameters are coupled, i.e., energy level of the nodes and communication range necessary to maintain connectivity. Given that sensor nodes are resource constrained, eventually a WSN has to consider the depletion of node batteries leading to the partitioning of the network. The energy dissipation, however, is generally spatially correlated. Therefore, groups of nodes form hotspots that deplete to coverage holes. A hole can be defined as a part of the network, which due to the energy depletion is no longer covered. These holes, when grow, can disconnect a part of network from accessing the sink, defined as a partition. If the network energy state can be modeled and predicted, then we can predict the occurrence of the holes and consequently the partitions. The holes and partitions appear as regions in an energy map. Our framework has all the necessary tools to profile the energy dissipation patterns, predict the network future energy state and detect the regions formed due to partitioning. Therefore, partition prediction becomes a natural candidate problem to be solved using our framework.

We can now define the problem according to the abstraction of our framework. A grid cell (cell in a grid map) gets disconnected from the network if it has energy below a minimum threshold so that it can not communicate anymore. These depleted grid cells form a region that represents a hole in an eMap. Partition, however, is a group of non-depleted grid cells that can not access sink due to the holes. It is therefore sufficient to profile the energy status of the network during its lifetime by collecting the energy profiles in order to predict network partitioning. As per definition the adaptation of the framework to predict network partitioning consists of three phases (Section 4) that we discuss as follows.

5.2 The Data Collection Phase

The nodes start the formation of 1-hops clusters as in (Tulone and Madden, 2006). During the 1-hop cluster formation cluster heads are selected, the cluster heads learn about their neighboring clusters. All cluster heads start aggregating the energy values. $AR3$ model is fitted and our proposed DRA (Alg. 1) is executed based on these models. The models for regions are aggregated using our proposed BTA (Alg. 2). Consequently, models are periodically sent to the sink.

5.3 The Prediction Phase

The sink regenerates the time series (energy values of cluster and hence the nodes) by applying reverse transformation. The data regeneration of the reporting nodes actually generates the energy profiles. The energy profiles of each node are modeled and predicted as described in Section 4.2. Energy dissipation is a decaying process so the time series contains trends but no seasonal components. The trends are removed by fitting polynomials. ARMA models are fitted to random components, selecting the best fit model using AIC criteria. After completion of modeling the node energy values are predicted and hence the future WSN energy profiles.

5.4 The Event (Holes and Partition) Detection Phase

The first step towards the abstraction of the WSN profile as a grid map (eMap in this case) is the selection of resolution, i.e., grid cell size at which this event (network partitioning or holes) is to be detected. From the formulation

of the problem we know that we have two coupled parameters, i.e., energy and communication range. Therefore, an upper bound for γ is the communication range (R). To accommodate a worst case scenario of two nodes lying on opposite corners of two grid cells, γ is given by $\gamma < R/2\sqrt{2}$, as shown in Fig. 3. The lower bound can be obtained from the node density, it should be selected such that the network area is not over sampled, as we show in simulation Section 6.2. A cell is connected to the neighboring cells until at least a single node has enough energy to communicate. The node having the highest energy level is selected as reporting node and Eq. (5) becomes $\xi_{ij} = max(v_n)\ \forall\ n \in g_{ij}$.

The predicted energy profiles are converted to the eMaps. CRA developed in Section 4.3 is used for both partition and hole detection on these eMaps. As per the given scheme we define two energy classes at 10% and below as the partition (or hole) class, and above 10% as non-partition class. This definition of energy classes gives the areas that are vulnerable to partitioning because of low energy.

6 Evaluation - Viability of our Approach

To evaluate how well our framework meets the design requirements, we use the problem of partition prediction as formulated in the case study. To determine accuracy and efficiency of the framework we compare the modeled and then regenerated energy values with the actual sensed energy values generated on the nodes during simulations. We predict the future energy states of every node and hence the future profiles of the network. The future profiles are then converted to maps. We denote these predicted maps as future grid maps Gf.

6.1 Evaluation Metrics

The transformation of a value spatial distribution into a map is a three stage process, i.e., a grid map, then a class map and finally a regions map. The regions map is physically same as a class map with additional information of region borders. Hence, we use two error criteria for the grid map and the class map. We use two more metrics to assess the accuracy of regions and efficiency in terms of number of packets generated for models in the network. Our first metric is the ***mean square error*** (Eq. (7.1)) between the reference grid map Gr (the actual data generated on the nodes) and the test grid map Gf , defined as

$$Ge = \frac{\sum_i \sum_j (\xi r_{ij} - \xi f_{ij})^2}{m}\{7.1\}, Ke = \sum_i \sum_j count(Kr_{ij} - Kf_{ij})\{7.2\} \quad (7)$$

where (i, j) are grid cell coordinates, Ge is the mean square sum of error, ξr_{ij} is the grid cell value of Gr and ξf_{ij} is the grid cell value of Gf and m is the number of occupied grid cells. Gr is the true data generated on the nodes, while Gf is the predicted map from the gathered data from network, which undergo local modeling and hence will deviate from true data due to modeling. Ge determines the relative accuracy of our approach against the ideal case.

The second metric ***misclassification cell count*** (Eq. (7.2)) counts the misclassified cells between the reference and the test class map. Ke is the total count of class cells that differ between the reference K_r and test class map Kf. 'count' function returns '1' if the two cells do not belong to the same class else it returns '0'. Ke is the direct measure of correct classification of the grid cells into the classes and indirect measure of the accuracy of area and border of the

detected event area. Our third metric is the misclassified cells percentage for each region to assess the accuracy the framework on regions level that we call ***regional percentile error***. The fourth metric ***message count*** is the efficiency metric, where we count the messages required to collect all the profiles of the network.

6.2 Simulation Settings

As two phases of the framework are carried out on the sink, therefore we performed our simulations in Matlab. It is a very well known simulation tool and suits our work as it facilitates to model energy dissipation patterns of very huge number of nodes. The network that we used in our simulations is generated as a random non-uniform distribution of nodes. The node distribution, as shown in Fig. 4, was selected to cover many possible scenarios in a real deployment. It contains some areas with high node density and some with low node density. It also contains two narrow bridges between two parts of the network that may lead to network partitioning. For energy dissipation modeling the common hotspot model (Zhao, 2002) was used. The energy dissipates in a spatially correlated manner around the hotspot. The nodes nearest to the hotspot are more active and hence dissipate more energy. The parts of the network that act as the coverage-bridge between two parts of the network and around the sink show relatively high energy dissipation rates. Subsequently these areas are modeled as hotspots.

We used a network containing 5000 sensor nodes that span in an area of 50×100 $unit^2$, each node having a communication range $R = 2$ units. For $R = 2$ the upper bound for grid cell size is 0.7 units. We found 0.3 as the lower bound because if we take a grid size smaller than 0.3 then we have more occupied grid cells than the number of nodes that over samples the network area. We therefore selected three grid sizes between upper and lower bounds 0.3, 0.5 and 0.7 units. Energy dissipation history of 164 profiles was collected from all the nodes. To evaluate the statistics we divided the history of profiles into two parts. 139 profiles were used for modeling purposes and 25 used for validation. 164 profiles represent the network lifetime history. If we scale 164 lifetime profiles to 164 days then 139 days of network operation are used to predict the next 25 days network status. 139 profiles of WSN were used to predict next 25 profiles. Each predicted profiles of the network was transformed to grid map.

6.3 Simulation Results

Fig. 5(a) shows the mean square sum of error for 25 prediction steps for 3 different grid sizes.The low values of mean square error show that the predictions are very accurate. The increasing trend is natural, as an increasing number of prediction steps makes the prediction model less accurate.

Fig. 5(b) shows the misclassified cells count. The mean square error results (Fig. 5(a)) imply that we can not expect much inaccuracy in misclassification. The highest count is naturally in the case of grid size 0.3, which reaches 88 at the peak. The total number of occupied grid cells at this resolution is 4196, so a worst case misclassification of 88 cells accounts to around 2% of the total cells. We also see a an increasing trend in the misclassification for each prediction step because of the increasing error between model approximation and the actual data. The oscillations in the graph give interesting insight. We have defined two classes of energy and as soon as the grid cells cross the class threshold (10% of energy) they

are classified into the partitioned class. The crests appear when cells in the actual data (Kr) cross the threshold of 10% but the cells from the modeled data (Kt), due to the lag in value, do not cross the threshold at the same time. Therefore, cells from the reference class are classified in the partitioned class but corresponding cells in the test class are still in the non-partitioned class, which increases the count of difference cells. As soon as the modeled data crosses the threshold the error decreases and troughs appear but a clear trend in increase of error continues.

Fig. 6 gives account of the error in the detected regions predicted through profiles of the WSN. To summarize the results we have selected prediction maps separated by five prediction steps. On first prediction step there are only three regions with less than 2% max percentile error. With each next prediction step the number of regions increases and errors distribute between the different regions. In the worst case scenario a region has a maximum percentile error of less than 2.7%. The results however show that each region is very accurately detected. Misclassification per region on the average is less than 3% which shows the accuracy of our approach to detect the regions and their boundaries.

Now, we summarize the results w.r.t. efficiency metric, i.e., number of packets needed to create energy profile of the whole network. To profile the whole network of 5000 nodes and to collect 164 profiles over the entire lifetime requires nearly 1 million data points. This overhead is reduced dramatically with the help of our DRA. Models are constructed for clusters and regions are formed based on the models using DRA. We fix the length of history that can be represented by a model. We chose 8, 14 and 20 values to be modeled by a single model. The graphs in Fig. 7 (a) and (b) represent cumulative sum of region formed and outliners respectively that we require over the lifetime of network using DRA. To observe the impact of ΔNc_i (the max: outliners allowed for DRA) we chose values from 0 to 4 to allow the clusters to form regions. As a model does not approximate 100% accurately all the data, therefore in addition to ΔNc_i, we allow outliners for each cluster head to build its model and fit to its data. Therefore, we will observe outliners even when we set $\Delta Nc_i = 0$. From Fig. 7(a) we can conclude that the number of regions being formed decreases as we allow more outliners. Allowing more outliners relaxes the neighboring clusters to fit to a given model. Similarly from Fig. 7(b) the number of outliners to be reported naturally increases also. The length of data to be represented by a model also affects the regions to be formed. The shorter length (8) forms more cumulative regions over the lifetime as it has to report more often than the longer length models that manage to fit more data within one model. However, shorter length model has to report very few outliners first, but shoots immediately as shorter length has more regions. Longest length models (20) have least regions initially again because it reports the complete data in less cycles of sending the models but increases to maximum as its very hard for the neighboring clusters to agree for a very long length of attribute. These long length models also have the maximum number of outliners to be reported even ΔNc_i=0. Therefore, maximum number of regions are formed but decreases afterwards as it has to make less updates. The data length to be modeled is a very important factor in this whole compression scheme. We found from our analysis that there is a compromise between the two extremes, which in our case was around 14. At this value the number of regions formed and the number of outliners to be reported is balanced between the two extremes.

Each region in Fig. 7(a) is equivalent to sending a message to the sink as a region is represented by this model. Three outliners are grouped in one packet, the number of packets for outliners is equivalent to $1/3rd$ of the outliners. With the settings of modeled data length=14 and ΔNc_i=1, we have to send 40499 packets, which is 4.93% of the total raw data to be reported otherwise.The results obtained in the evaluation are in accordance with the design requirements of the framework. It is lightweight as DRA and BTA cumulatively reduce the data to less than 5% of the raw data needed to profile all nodes. The achieved predictions are long-term and accurate, represented by the maximum prediction error of approximately 3% in misclassification of the regions of the map for 25 prediction steps. CRA detect energy holes that will partition the network in 22nd days (in scaled time as explained in Section 6.2). From Fig. 6, we conclude that the partition prediction is more than 97% reliable (because of 97% region accuracy).

7 Conclusion and Future Directions

We developed a generalized framework for efficient predictive monitoring to forecast events in order to support an autonomic self* system for WSN. We demonstrated that it can be effectively used to predict events related to different attributes. We described it as a three phase strategy. In the data collection phase we proposed efficient algorithms to spatio-temporally compress the attribute values and transport them to the sink. In the prediction phase, we long-term predicted the attribute states. In the event detection phase, we proposed a generalized event detection algorithm. We demonstrated the feasibility and validity of approach by predicting the network partitioning as a case study. We were able to predict multiple holes and the resulting partitioned area of the network; information necessary to initiate proactive self* actions. Simulations support the practicality of our approach by showing its high accuracy and low monitoring overhead on the network. In order to further increase the efficiency, we propose to adapt the spatio-temporal data compression to the occurrence probability of events. For instance sensor nodes should increase data model accuracy if they are located in areas where events frequently occur or when an event is suspected. We also plan to extend our approach for proactive reconfiguration of network entities to enhance functionality and dependability through the predicted events.

References

F. Aurenhammer. Voronoi diagrams - A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

T. Banerjee et al. Fault tolerant multiple event detection in a wireless sensor network. *J. Parallel Distrib. Comput.*, 68(9):1222–1234, 2008.

B. Gedik et al. ASAP: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Dist. Systems*, 18(12):1766–1783, 2007.

T. He et al. Range-free localization and its impact on large scale sensor networks. *Trans. on Embedded Computing Sys.*, 4(4):877–906, 2005.

O. Landsiedel et al. Accurate prediction of power consumption in sensor networks. In *IEEE Workshop on Embedded Networked Sensors*, pages 37–44, 2005.

L. Ljung. *System Identification: Theory for the User (2nd Edition)*. Prentice Hall PTR, December 1998. ISBN 0136566952.

R. A. F. Mini et al. A probabilistic approach to predict the energy consumption in wireless sensor networks. In *In IV Workshop de Comunicao sem Fio e Computao Mvel. So Paulo*, pages 23–25, 2002.

D. C. Montgomery et al. *Introduction to Time Series Analysis and Forecasting*. Wiley, 2008.

S. Rost and H. Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks (SECON), 2006. Third Internatinal Conference on*, volume 2, pages 575–584, 2006.

F. K. Shaikh et al. TRCCIT: Tunable Reliability with Congestion Control for Information Transport in Wireless Sensor Networks. In *International Wireless Internet Conference (WICON)*, (to appear), 2010.

K.-P. Shih et al. PALM: A Partition Avoidance Lazy Movement Protocol for Mobile Sensor Networks. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2484 – 2489, 2007.

N. Shrivastava et al. Detecting cuts in sensor networks. In *Proc. of the 4th int. symposium on Info. processing in sensor networks (IPSN)*, pages 210–217, 2005.

I. Solis and K. Obraczka. Isolines: energy-efficient mapping in sensor networks. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 379 – 385, 2005.

D. Tulone and S. Madden. Paq: Time series forecasting for approximate query answering in sensor networks. In *European Conf. on Wireless Sensor Networks (EWSN)*, pages 21–37, 2006.

X. Wang, J.-J et al. Robust forecasting for energy efficiency of wireless multimedia sensor networks. *Sensors*, 7(11):2779–2807, 2007.

W. Xue, Q. Luo et al. Contour map matching for event detection in sensor nets. In *Proc. of the Int. Conf. on Management of data (SIGMOD)*, pages 145–156, 2006.

J. Yick et al. Wireless sensor network survey. *Comput. Netw.*, 52(12):2292–2330, 2008.

L. Yu et al. Real-time forest fire detection with wireless sensor networks. In *Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM).*, volume 2, pages 1214–1217, 2005.

Y. J. Zhao. Residual energy scan for monitoring sensor networks. In *Wireless Communications and Networking Conference (WCNC)*, pages 356–362, 2002.

M. Mamei and R. Nagpal. Macro programming through bayesian networks: Distributed inference and anomaly detection. In *Proc. of the Fifth IEEE Int. Conf. on Pervasive Comp. and Comm. (PERCOM)*, pages 87-96, 2007.

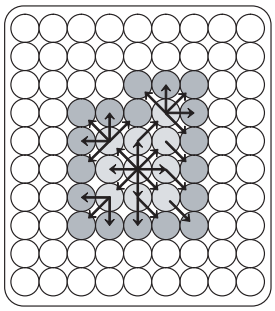


Figure 1 Execution of DRA

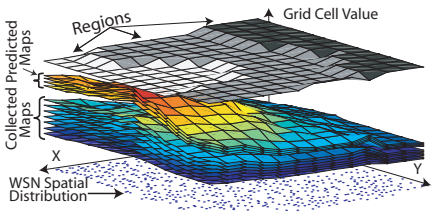


Figure 2 Temporal stack of the grid maps

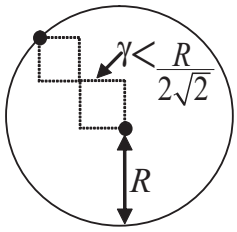


Figure 3 Max grid size

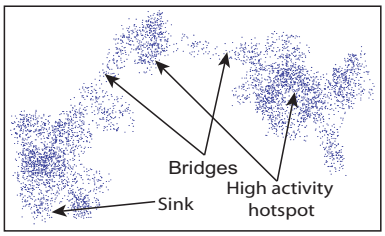
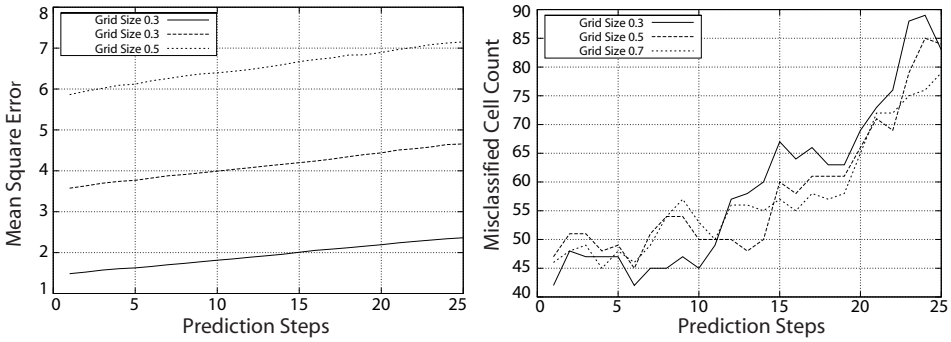


Figure 4 Node distribution



(a) Mean square error (b) Misclassification cell count

Figure 5 Predictability and accuracy measures of the framework

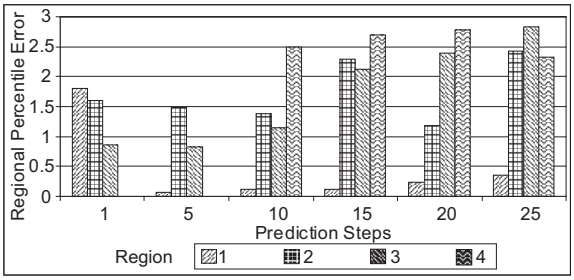
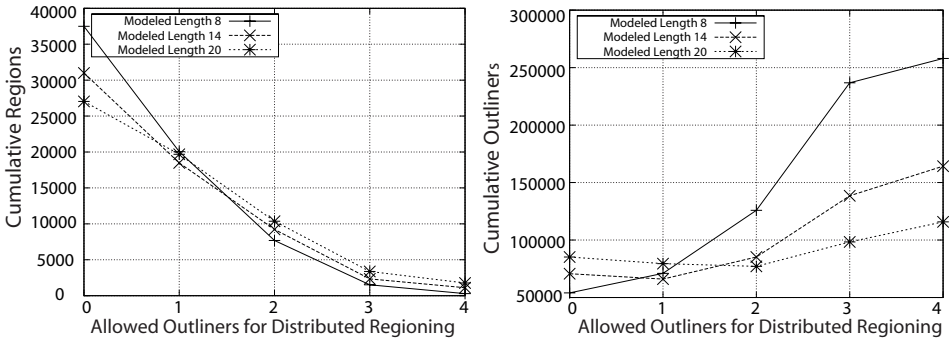


Figure 6 Misclassification percentile error per region



(a) Cumulative Regions (b) Cumulative Outliners

Figure 7 Regions and outliners for data compression using DRA