MPM: Map based Predictive Monitoring for Wireless Sensor Networks[⋆]

Azad Ali, Abdelmajid Khelil, Faisal Karim Shaikh, and Neeraj Suri

Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany.
{azad,khelil,fkarim,suri}@informatik.tu-darmstadt.de

Abstract. We present the design of a Wireless Sensor Networks (WSN) level event prediction framework to monitor the network and its operational environment to support 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. We propose a Map based Predictive Monitoring (MPM) approach where a selected WSN attribute is first profiled as WSN maps, and based on the maps history, predicts future maps using time series modeling. The "attribute" maps are created using a gridding technique and predicted maps are used to detect events using our regioning algorithm. The proposed approach is also a general framework to cover multiple application domains. For proof of concept, we show MPM's enhanced ability to also accurately "predict" the network partitioning, accommodating parameters such as shape and location of the partition with a very high accuracy and efficiency.

Key words: Predicitve Monitoring, Time Series Analysis, Wireless Sensor Networks, Event Prediction

1 Introduction

Wireless Sensor Networks (WSN) typically entail an aggregation of sensing/ 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 often used for monitoring of spatially distributed attributes such as detection of physical events, e.g., fire, temperature gradients and high/low pressure. To maintain the required WSN dependability, network events such as partitioning of the network are also to be detected. The reporting of such events beyond simple monitoring becomes highly useful if these events can be predicted in advance. Consequently, appropriate autonomic actions could be taken either to avoid or delay events from happening by triggering self* actions.

Varied works exist for event detection [1] such as fire detection [2] and network partitioning [3–5]. Most of these efforts are specialized for specific scenarios.

⋆ Research supported in part by HEC, MUET, EC INSPIRE, EC CoMiFiN, and DFG GRK 1362 (TUD GKMM)

Some consider generic scenarios [6], but they suppose the event to take certain shapes and patterns. Also most efforts focus on detecting the events after they have already happened. In the perspective of reacting to the event, it may be too late to detect the *events* after it has already taken place. It would be far more practical if we could predict the event occurrence.

Multiple efforts exists for predictions [7, 8, 11–13]. However, most are either limited to predicting specifically a certain attribute like energy or provide only node level short-term prediction for data compression to minimize data to be reported from the network.

It is natural to combine prediction of physical and network events with generalized event detection to take proactive actions. To the best of our knowledge there exists no work that proposes generalized event prediction. In this paper we develop a framework to predict the future states of the network for the attribute of interest such as temperature or residual energy. Based on the predicted states of the network we develop a generalized event detection technique. In this work we target long-term predictions that require more computational resources than a sensor node has and sufficient history of the attribute that contains all variation patterns. Hence, we collect many profiles of the network and perform the modeling operations on the sink. Typically, the sink uses optimized data collection techniques to collect such a history of the interested attributes from the network [8]. We refer to the collection of attribute values or samples from network as *profiling*.

In the WSN environment, events usually are defined as spatially correlated attribute distributions [6]. Thus, the spatial distribution of attribute of interest needs to be quantified in order to detect events. One such natural spatial quantification is the *Map*. For a WSN an eMap is an energy map that represents the current residual energy of the network [9], or tMap for temperature etc. Maps can generally be created for any attribute and provide a basic utility to detect events using pattern matching [6]. The use of maps in our work emphasizes the fact that there is a wide class of events in which a discrete view of a network on node level is either not necessary or not efficient. Rather, a view at a higher abstraction level of regions and maps, that represent a group of nodes, is needed. For example in temperature, pressure, humidity and residual energy monitoring a map is far more meaningful than discrete node values. This also relates to the inherent WSN node redundancy that leads to spatially correlated node states. On this background this paper presents three specific contributions, namely

– Design of a generalized framework for sink aided profile prediction. The framework is adaptable to different simple as well as complex physical and network events.
– Development of a regioning technique to detect events from predicted profiles to support autonomous actions in the specified regions of the network.
– As validation for our Map based Predictive Monitoring (MPM) approach, we propose a solution of network partition prediction as a case study.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Section 3 gives the preliminaries. Section 4 details our MPM approach for predictive monitoring in WSN. In Section 5, the case study is given. In Section 6, we evaluate our approach for the given case study through simulations. Section 7 concludes our work and outlines future directions.

2 Related Work

A variety of work is available for event detection [1]. The most relevant work to our event detection strategy is [6] that investigates map based event detection and 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. They further define three common types of events namely Pyramid, Fault and Island. Our work is independent of event shape due to our regioning technique. Furthermore, our framework predicts the events rather than just detecting it. In [10], Banerjee et al. present 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 like partition detection [3], fire detection [2] and others [1]. These specific solutions do not feature portability to adapt to different application scenarios.

There is a variety of work to monitor WSN's and to minimize the overhead of data collection for the monitoring. Strategies in [7] and [11] predict the power consumption in WSN. In [12] Mini et al. propose a network state model and use it to predict the energy consumption rate and construct the energy map accordingly. In [13], the authors focus on predicting energy efficiency of multimedia networks. These works concentrate on predicting specifically the energy, also they do not provide any extension of their work to attributes other than energy. Authors in [14] give a theoretical framework for abstracting the world as WSN maps. Our work in this paper however presents a practical approach not only for abstraction but also facilitates to predicts the events to take place in future.

As we present a case study for partition prediction, hence here we discuss the related work in this respect. In [3], this problem has been addressed for a sub-class of linearly separable partitions, i.e., cuts. Memento [4] is a health monitoring system that suggests to continuously collect connectivity information at the sink to be able to detect network partitioning. The Partition Avoidance Lazy Movement protocol for mobile sensor networks [5] is a decentralized approach, where a sensor node can locally suspect network partitioning and move to avoid it. 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. Based on our event prediction framework as an example we propose a solution that is generalized and is not dependent on the shape, size or location of the partition. Moreover, we provide prediction of the time, when network partitioning is going to happen.

3 Preliminaries

We now describe the system model, the requirements driving our approach and give basic definitions.

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 [15] or GPS. A typical WSN deployment may contain hundreds of sensor nodes with varying densities according to the coverage requirements. In this work, we do not consider a particular node distribution. We assume all sensor nodes to be homogeneous. Hence the nodes have the same transmission range R and same initial battery capacity. We consider that nodes crash due to energy depletion only. We assume the events for predictions to be happening over a longer period of time, for example, events that may take hours, days or even months to develop. We consider that events are not spontaneous, spatially correlated, do not depend discretely on a single node and are predictable.

3.2 Requirements on the MPM

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

3.3 Definitions

Here we give some basic definitions that are necessary to develop the MPM framework.

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,*

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

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

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

Definition 5 *The quantification of WSN space and the conversion of a WSN to a map level abstraction is the key to detecting generic events. The abstraction of WSN as map, transforms the event into a region of a map as shown in Fig. 1. For example an event of fire will be a region of a map in which the value of temperature exceeds a given threshold. In our framework we define an event as a region of map whose values fall in the range of attribute values for which event is defined.*

4 Predictive Monitoring: The MPM Approach

We present here our MPM Framework that can be used to support self* actions. To keep the event prediction as generic as possible, we have proposed it as a four phases process. In each phase we have proposed a technique that is independent of the attribute to be monitored. It is important to highlight that the use of proposed techniques in the framework does not imply to limit the framework to only these, rather for a particular implementation specialized techniques can always be easily accommodated due to its modular structure. The four phases of the framework are summarized here. *The segmentation* phase specifies the properties of the grid maps (Def. 4) such as grid cell size. In *the data collection* phase, data is periodically but efficiently fetched from the network on the sink. *The prediction* phase is used for predicting future status of the network in the form of future grid maps. *The event detection* phase is used to detect events (Def. 5) in the predicted grid maps. These phases are individually detailed in the following sections.

4.1 The Segmentation Phase

In order to reach an acceptable spatial resolution with higher level abstraction of network as a map, we considered virtual grids and Voronoi diagram [16] techniques to segment (Def. 4) WSN space. 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 allows to investigate prediction accuracy and profiling efficiency tradeoffs. Grid allows such flexibility therefore, we base our segmentation on grid.

The virtual grid or simply *grid* divides the WSN area into fixed size squares or grid cells as shown in Fig. 1. Thus nodes that fall within a cell are grouped.

For the grid maps construction, two parameters must be specified. The first parameters 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. Depending on the application the appropriate value of γ is affected by (1) physical parameters such as attribute's spatial distribution, (2) network parameters, such as communication range, (3) application requirements such as the (zoom) level at which to detect the event. In applications such as temperature and humidity, the grid size can be selected big enough that it represents the patches of the geographic areas, each differing considerably in the attribute values.

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 to sense 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

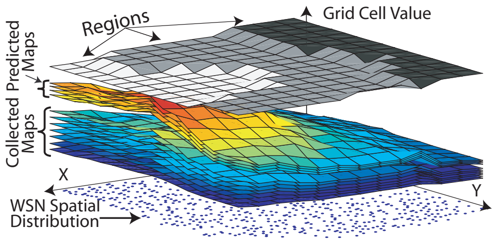


Fig. 1: Temporal stack of the grid maps

$$\xi_{ij} = f(v_n)\ \forall\ n \in g_{ij} \tag{2}$$

During map construction the nodes undergo coordinate transformation. If a node has $(x,\ y)$ coordinates then the grid cell coordinates $(i,\ j)$ can be calculated as

$$i = \lfloor x/\gamma \rfloor, j = \lfloor y/\gamma \rfloor \tag{3}$$

All the nodes in the area of a certain grid cell have same grid cell coordinates.

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.

4.2 The Data Collection Phase

In this phase the profiles are obtained by collecting the attribute values from the network for which the event is to be predicted. The grid technique also aids in reducing the number of the reporting nodes by forming clusters of nodes. Once the grid size has been specified (and disseminated) in the network, the nodes calculate their corresponding grid cell using coordinate transformation. Consequently, in-network aggregation can be performed according to f by the nodes within the same grid cell. Only single node should then report ξ to the sink. This reduces the amount of data to be reported to the sink enormously.

Simple criteria can be used for selection of reporting node such as, node with maximum energy. It is not efficient even for reporting nodes to send the ξ values periodically to the sink. To further reduce the overhead on the network as per our design requirements, the aggregated attribute are modeled as in [8] and only the model parameters are sent to the sink, which can be used to regenerate the actual data. The model is recalculated only when the prediction error exceeds a predefined threshold. With each model update the outliners of the last model are also reported. To achieve this the reporting nodes maintain a limited history of the aggregated attribute values to be reported to the sink and fits 3^{rd} order autoregressive model ($AR3$) which is only a particular case of the $ARMA(p, q)$ model (Def. 3), when $p = 3$, $q = 0$. This data compression approach uses fixed simple model because nodes have too low computational resources to determine the parameters of a general ARMA model.

As per requirement on our framework we emphasize the generality of MPM with respect to data collection, since we abstract the attribute type as a generic time series (Def. 1). Using this technique any attribute type can be reported to sink with a comparable efficiency, making our framework independent of the attribute type to be reported. We fulfill our first requirement of framework to be lightweight using the segmentation and data collection techniques. These techniques collectively reduce the overhead on the network tremendously.

4.3 The Prediction Phase

The models received on the sink in data collection phase from each reporting node are used to regenerate the variation patterns or attribute history through reverse transformation. The regenerated history is essentially the grid map representation of the WSN. This forms a temporal stack of the grid maps as shown in Fig. 1. Each grid cell in the grid maps stack can be treated and modeled as a separate time series for prediction. Individual models of each grid cell can then be used to predict future values by fitting a prediction model, effectively predicting grid maps. The time series can be modeled in different ways [13]. In this paper, we use the widely used time domain modeling because of its general applicability.

It is important to point out that time series modeling performed in this section is different from that done on node level for data collection, which is only short-term prediction to compress the data using the fixed $AR3$ model. In this phase, we perform a full scale modeling of the collected data to predict the future states using the complete history and model each component separately.

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

$$X(t) = T_t + S_t + R_t \tag{4}$$

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 Box-Jenkins Model to model a time series containing any of these components.

Box-Jenkins (BJ) Model: Box-Jenkins model predicts a time series by fitting it an ARIMA process (Autoregressive Integrated Moving Average). 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. 5.1) or Moving Average (MA, Eq. 5.2)

$$X(t) = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} \ \{5.1\}, \ X(t) = \theta_1 Z_{t-1} + \cdots + \theta_p Z_{t-q} \ \{5.2\} \quad (5)$$

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: As Box-Jenkins model requires a time series to be stationary (Def. 2). 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

$$\rho_k = \frac{\sum_{i=1}^{N-k} \left(X_i - \bar{X}\right)\left(X_{i+k} - \bar{X}\right)}{\sum_{N}^{i=1} \left(X_i - \bar{X}\right)^2} \quad (6)$$

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) [17]. To automate the model selection process either Akaike's Information Criterion (AIC) or Akaika's Final Prediction Error (FPE) [18] 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 [18].

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.

To fulfill our second requirement of long-term prediction and to accommodate any variation patterns in the time series we use a generic time series modeling techniques that models each component (trend, seasons, random) to facilitate long-term predictions. We keep the generality of framework valid also in this phase by abstracting the attribute to be predicted as time series, which makes this phase also independent of attribute type.

4.4 The Event Detection Phase

From Def. 5 we know that events appears as regions in a map. We introduce here a generalized regioning technique that can detect the regions and their perimeters which leads to generic event detection.

The regions are formed due to the fact that the attribute values fall into 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 such as temperature above 90^o for fire). This gives us a more acceptable abstraction than the exact values themselves. Therefore the 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 C as c_1, $c_2, \cdots$ 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

$$C = \begin{cases} c_1 & \text{if } \xi_2 < \xi_{ij} \leq \xi_1 \\ c_2 & \text{if } \xi_3 < \xi_{ij} \leq \xi_2 \\ \cdots \end{cases} \tag{7}$$

Our regioning algorithm (Fig. 2) takes the grid map as input and determines perimeter and regions belonging to different classes and hence events. We refer to the resultant output as the *regions map*. The grid cells in the grid map that belong to the same class are grouped to form the regions. The regioning technique 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. 7. 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 the regioning algorithm independent of the shape that a region takes or the number of regions (hence the number of events) in the map.

```
(mapX, mapY)=dimensions(map);
regionsMap[]= -1;
mapBorders[][];
while there is cell not assigned region yet do
    regionBorder[]= (Find cell with regionsMap=-1)
    dilateRegion(map,regionBorder[],regionsMap[],regionId)
    mapBorders[regionId][]=regionBorder;
    regionId++;
end while
dilateRegion(map,regionBorder[],regionsMap[],regionId)
repeat
    changeInBorder=0;newRegionBorder[]=0;
    for (currentCell=each cell in regionBorder[]) do
        neighborsList[]= eightNeighborsOf(currentCell);
        for each neighbor in neighborsList[] do
            if (currentCell & neighbor are in the same class)
            & neighbor in regionsMap[]==-1 then
                neighbor in regionsMap[]=regionId;
                include cell in the newRegionBorder[];
                changeInBorder=1;
            end if
        end for
        if currentCell and (1 or more) neighbors are not
        in the same class then
            add cell to newRegionBorder[];
        end if
    end for
    regionBorder[]=newRegionBorder[];
until changeInBorder
```

Fig. 2: Regioning algorithm

The algorithm starts by defining all the cell as *not assigned a region* by initializing the variable regionsMap[]=-1. The algorithm next 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 (line 5, cell with -1 in regionsMap is not assigned a region yet). Algorithm then starts expanding/*dilating* the region (line 6). To expand the region, the neighboring eight cells around the this region cell are checked if they already belong to the any class (line 14-16), if not then they are also classified according to Eq. 7. 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 (line 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 (line 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 (line 4,9).

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

5 Case Study: MPM Adaptation for Predicting Network Partition

To use our framework for network partition prediction, the problem needs to be formulated according to the abstractions (maps, classes etc.) in the framework.

5.1 Problem Formulation

Partition detection is a complex problem as a physical and a network parameter are being 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 can sometimes 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 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 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 eMaps in order to predict network partitioning. As per definition the adaptation of the MPM framework to predict network partitioning consists of four phases that we discuss as follows.

5.2 The Segmentation Phase

The first step towards the abstraction of the WSN network 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 the $\gamma < R/2\sqrt{2}$, 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.

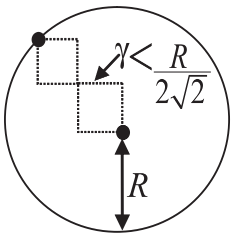


Fig. 3: Max grid size

5.3 The Data Collection Phase

Once the grid cell size is specified, the nodes determine their corresponding grid cells using Eq. 3. As we discussed earlier energy dissipates in a spatially correlated manner, the hotspot energy dissipation model is considered [9]. Therefore, the nodes in a grid cell are expected to have similar energy dissipation pattern. A cell is connected to the network 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. 2 becomes

$$\xi_{ij} = max(v_n) \; \forall \; n \in g_{ij} \tag{8}$$

All reporting nodes start aggregating the energy values. $AR3$ model is fitted to the data as per the scheme given in Section. 4.2. Model parameters are sent to the sink. The sink regenerates the time series (data of reporting sensor node) by applying reverse transformation. The regeneration of the data of the reporting nodes actually generates the grid maps according to the given parameters.

5.4 The Prediction Phase

The time series of each grid cell is modeled and predicted as described in Section. 4.3. 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 grid cell values are predicted and hence the future grid maps.

5.5 The Event (Holes/Partition) Detection Phase

The regioning algorithm developed in Section. 4.4 is used for both partition and hole detection. 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. The regioning algorithm detects these regions along with the perimeter. The regions with energy 10% and below are holes in the network. To detect partitioning however, the algorithm does not need to find all the regions, therefore it executes only two iterations. In the first iteration with two energy classes it starts from the area connected to the sink and merges the non partitioned area around it. In the next iteration the classes are omitted and rest of the area is merged to find its perimeter that represents the partitioned region.

6 Evaluation - Viability of our Approach

To evaluate how well our framework meets to design requirements, we evaluate it for the problem of partition prediction as formulated in the case study. To determine accuracy and efficiency of MPM we compare it with ideal case situation in which the data from all nodes is assumed. In the ideal case we predict the future energy states of every node separately and hence the future profiles of the network. The future profiles are then converted to maps. We denote these maps predicted using profiles of all the nodes as ideal grid maps Gi. We denote maps generated through our approach as optimized grid maps Go.

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 perimeters. 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 fetched packets from the network. Our first metric is the ***mean sum of absolute error*** *(Eq. 9.1)* between the reference grid map Gr (the actual data generated on the nodes) and the test grid map (Gi and Go), defined as

$$Ge = \frac{\sum_i \sum_j abs(\xi r_{ij} - \xi t_{ij})}{m}\{9.1\}, Ce = \sum_i \sum_j count(Cr_{ij} - Ct_{ij})\{9.2\} \quad (9)$$

where (i, j) are grid cell coordinates, Ge is the mean sum of absolute error, ξr_{ij} is the grid cell value of reference map and ξt_{ij} is the grid cell value of test grid map and m is the number of occupied grid cells. Gr is the true data generated on the nodes, while Gi and Go are 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. 9.2) counts the number of misclassified cells between the reference and the test class map. Ce is the total count of class cells that differ between the reference Cr and test class map Ct (Ct are ideal class map Ci and optimized class map Co). 'count' function returns '1' if the two cells do not belong to the same class else it returns '0'. Ce is the direct measure of correct classification of the grid cells into the classes and indirect measure of the accuracy of area and perimeter 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 profile the network.

6.2 Simulation Settings

As three phases of the framework are carried out on the sink, therefore we performed our simulations on 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 [9] 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.

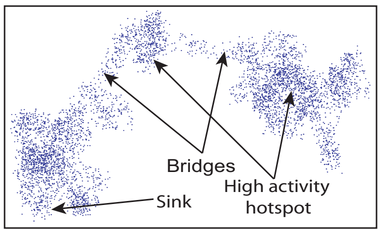


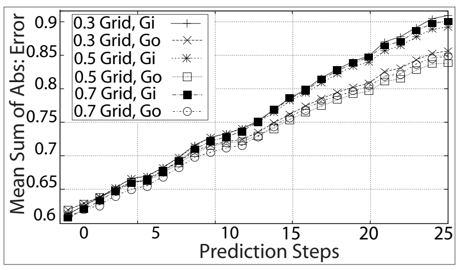
Fig. 4: Node distribution

We used a network containing 5000 nodes that span an area of 50×100 $unit^2$, with 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 for the ideal case from all the nodes and then using our approach 164 grid maps were collected from a subset of reporting 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. First, we considered the ideal situation and 139 profiles of each node were used to predict next 25 states. Each ideally predicted profile of the network was transformed to grid map. Then using MPM approach 139 collected grid maps were used to predict next 25 grid maps.

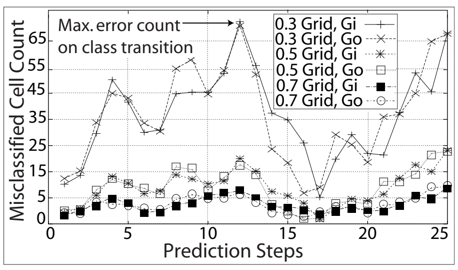
6.3 Simulation Results

Fig. 5(a) shows the graphs for mean sum of absolute error for 25 prediction steps for 3 different grid sizes. It shows that the optimized grid maps are almost as accurate as ideal grid maps. The mean error ranges from 0.6 to 0.9 even after 25 prediction steps, showing the level of accuracy of the prediction model. The lower bound of error 0.6 is actually the maximum error that was allowed in the local models on sensor nodes. The increasing trend is natural, as an increasing number of prediction steps makes the prediction model less accurate. In ideal case due to the cumulative error of considerably more number of nodes, the error is slightly more than our approach.

Fig. 5(b) shows the misclassified cells count. The results of the mean sum of absolute error imply that we can not expect much inaccuracy in misclassification graphs. The highest count is naturally in the case of grid size 0.3, which touches 67 at the peak. The total number of occupied grid cells at this resolution is 4146, so a worst case misclassification of 67 cells accounts to less than 2% of the total cells. We also see a slight increasing trend in the misclassification for each prediction step because of the increasing error between model approximation and the actual data. The peak in the graph gives interesting insight. We have



(a) Mean sum of absolute error

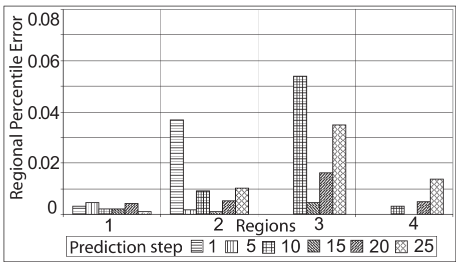


(b) Misclassification cell count

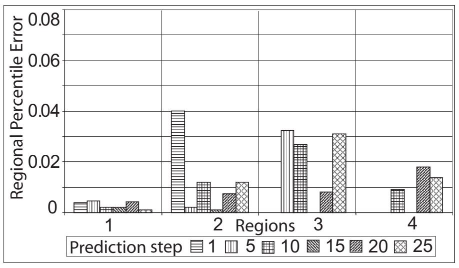
Fig. 5: Predictability and accuracy measures of MPM

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. From Fig. 5(a) it is obvious that there is a minimum difference of 0.6 between the reference data and test data, which creates a lag in the value of both data sets. This peak appears when some cells in the actual data (Cr) cross the threshold of 10% but the cells from the modeled data (Ct), due to the lag in value do not cross the threshold at the same time. Therefore, many 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 this peak disappears but a slight trend in increase of error continues.

Fig. 6(a) gives account of the error in the detected regions predicted through profiles of the nodes and Fig. 6(b) is the regional error predicted through collected maps. To summarize the results we have selected prediction profiles/maps separated by five prediction steps. On first prediction step there are only two regions with less than 0.04% 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 0.06%. The results however show that each region is very accurately



(a) Profile based prediction



(b) Map based prediction

Fig. 6: Misclassification percentile error per region

detected. Moreover our approach of maps that uses only a subset data is almost as good as the profiles that consists of the data from all the nodes. Misclassification per region on the average is less than 0.03% 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 for profiling. To profile the whole network of 5000 nodes and to collect 164 profiles for the entire lifetime, requires nearly 1 million data points. This overhead is reduced dramatically by sending the models instead of raw samples. It is further reduced to 14214 packets with the utilization of the gridding technique. This equates to each node sending less than 3 packets instead of 164, which is less then 1.8% of raw data to be collected. If the network is scaled to 100 nodes the packets to be sent scale down to 284. These results satisfy our requirement on the framework to be lightweight.

6.4 Discussion

The results obtained in the evaluation are in accordance to the design requirements for the framework. It is very lightweight as data compression and gridding cumulatively reduce the data to less than 1.8% of the raw data needed to profile all nodes. The prediction is very accurate, represented by max prediction error of approximately 0.06% in misclassification of the areas of the maps for 25 prediction steps. The 22 days (in scaled time as explained in Section. 6.2) earlier prediction of partition is also feasible for proactive self reconfiguration, enabling autonomicity of WSN.

Our framework ensures reliably accurate information for proactive action and well before the event takes place. The proactive action can be triggered using regioning algorithm results. If there exists an event region, it will be detected by the regioning algorithm along with the perimeter of the event regions. The regioning algorithm determines the perimeter of the holes and the partition, it implicitly provides the exact information about the area, location and the affected nodes that lie within that perimeter. As soon as the sink successfully detects an event, it can either trigger an early warning or initiate a proactive action like move some nodes to affected areas, redeploy new nodes etc. if applicable. The MPM framework gives vital information beforehand to act proactively, but this proactive action itself is beyond the scope of this work.

7 Conclusion and Future Directions

We have developed Map based Predictive Monitoring, a generalized framework for event prediction to support an autonomic self* system for WSN. To demonstrate the feasibility and validity of approach we predicted the network partitioning as a case study. We were able to detect multiple holes and resulting partitioned area of network; information necessary to initiate proactive self reconfiguration. Simulations support the practicality of our approach by showing its high accuracy and low monitoring overhead on the network. We plan to extend our approach for proactive reconfiguration of network entities to enhance functionality and dependability through the predicted events.

References

1. Yick, J. et al.: Wireless sensor network survey. Computer Networks, 52(12), 2292–2330 (2008).
2. Yu, L. et al.: Real-time forest fire detection with wireless sensor networks. In: WCNM, 2, pp. 1214–1217 (2005).
3. Shrivastava, N. et al.: Detecting cuts in sensor networks. In: IPSN, pp. 28 (2005).
4. Rost, S. and Balakrishnan, H.: Memento: A Health Monitoring System for Wireless Sensor Networks. In: IEEE SECON, pp. 575–584 (2006).
5. Shih, K. P. et al.: PALM: A Partition Avoidance Lazy Movement Protocol for Mobile Sensor Networks. In: Proceedings of the IEEE WCNC, pp. 2484 – 2489 (2007).
6. Wang, X. et al.: Contour map matching for event detection in sensor networks. In: SIGMOD, pp. 145–156 (2006).
7. Achir, M. and Ouvry, L.: Power consumption prediction in wireless sensor networks. In: 16th ITCS (2004).
8. Tulone, D. and Madden, S.: PAQ: Time series forecasting for approximate query answering in sensor networks. In: EWSN, pp. 21–37 (2006).
9. Zhao, J. et al.: Residual energy scan for monitoring sensor networks. In: WCNC, pp. 356–362 (2002).
10. Banerjee, T. et al.: Fault tolerant multiple event detection in a wireless sensor network. Journal of Parallel and Distributed Computing, 68(9), 1222–1234 (2008).
11. Landsiedel, O. et al.: Accurate prediction of power consumption in sensor networks. In: EmNets, pp. 37–44 (2005).
12. Mini, A. F. et al.: A probabilistic approach to predict the energy consumption in wireless sensor networks. In: IV Workshop de Comunicao sem Fio e Computao Mvel. So Paulo, pp. 23–25 (2002).
13. Wang, X. et al.: Robust forecasting for energy efficiency of wireless multimedia sensor networks. Sensors, 7(11), 2779–2807 (2007).
14. Khelil, A. et al.: MWM: A map-based world model for event-driven wireless sensor networks. In: Autonomics, pp. 1–10 (2008).
15. He, T. et al.: Range-free localization and its impact on large scale sensor networks. Transaction on Embedded Computing Systems, 4(4), 877–906 (2005).
16. Aurenhammer, F.: Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Computing Surveys, 23(3), 345–405 (1991).
17. Montgomery, D. C. et al.: Introduction to Time Series Analysis and Forecasting. John Wiley and Sons, New Jersay (2008).
18. Ljung, L.: System Identification: Theory for the User (2nd Edition). Prentice-Hall, New Jersey (1998).