CHAPTER 4

A COMMUNICATION EFFICIENT FRAMEWORK FOR DATA GATHERING SENSOR NETWORKS

4.1 INTRODUCTION

Chapter 3 discussed about the formation of a basic communication infrastructure that can be used to prolong the stability period of a wireless sensor network. This chapter discusses about how the energy expenditure can be further economized by means of reducing the amount of data reported by the sensor nodes. The most energy intensive operation that a node performs is communication. The cost of transmitting a bit is much higher than computing (Pottie and Kaiser 2000). The network attempts to minimize the energy expenditure either by minimizing the communication or by turning off the transceiver when no communication needs to occur. This chapter focusses on the problem of how the energy expended in continuously reporting the vast quantity of raw sensor data from numerous sensor nodes can be minimized.

The sensory data collected from the physical environments depicts a huge amount of correlation in both space and time domains. In this thesis, it is proposed to exploit these spatial-temporal correlations in the observed feature to address the aforementioned problem. In this thesis, an overlay clustering framework is proposed that partitions the physical sensor network topology into a number of feature regions consisting of nodes observing similar data patterns. A designated node in each feature region then
communicates only the model coefficients to its CH. These model coefficients are used by the CHs to answer queries from the base station. As long as the spatial stability exists in a given region, no new communication needs to be initiated from that region. This eliminates the need for periodically sending across a vast quantity of raw sensor data from numerous sensor nodes. This huge savings in communication cost is achieved by leveraging the processing potential available in individual sensor nodes.

Besides the need for saving communication cost, applications such as contaminant monitoring, tracking soil moisture level, water temperature estimation of river beds etc. require an understanding of the graphical delineation of features of interest in a monitored terrain (Singh and Prasanna 2005). Hence, given the close association of sensors to the physical world, grouping nodes according to their spatial attributes would be beneficial from multiple aspects. Here, sensor measurements are used as actual inputs to the problem. Therefore, one is able to achieve savings in data aggregation and communication costs, if as much redundancy as possible can be eliminated in the lowest possible hierarchy (Tulone 2006). More specifically, the objective of the reported work is to spatially cluster regions of the network so that the nodes inside each cluster region have high correlation in sensor measurements.

In the proposed architecture, the designated nodes alone communicate with the CH and the model coefficients are transmitted instead of the raw data; this approach dramatically reduces the amount of communication required to closely monitor the geographical region of interest. This enormous saving in the communication cost is achieved at a slightly higher processing complexity at the individual nodes. Nevertheless, this higher processing complexity is justified because a sensor node not only possesses the communication ability but also the computing ability; each node
is smart enough to prepare a local sketch of the observed data with the help of its embedded processor.

4.2 RELATED WORK

Anand et al (2002) proposed an in-network data aggregation approach for aggregating spatial data at multiple resolution points. But they did not deal with how feature regions can be constructed and maintained. Singh et al (2004) addressed the problem of building topographic maps and Singh and Prasanna (2005) demonstrated how the information stored in topographic maps can be used for efficiently routing and resolving topographic queries.

The energy efficient communication module proposed in this thesis is the extension of the work done by Meka and Ambuj (2006) and Tulone (2006). The former proposed the idea of spatially clustering regions of the network with similar trends and patterns. The latter proposed a technique of using the AR (auto regression) models to approximately answer queries based on time series forecasting; this idea is used in this thesis to compute the AR model coefficients which serve as the local drafts of the observed data at each node. The cluster head then uses these AR model coefficients to predict the sensor readings without directly communicating to the members. In this thesis, the minimum spanning tree clustering technique is adopted for the formation of feature regions.

4.3 FORMATION OF FEATURE REGIONS

In order to discover the global temporal-spatial patterns in the sensor data of the monitored region, feature regions should be constructed based on the trend behavior of the observed data. Therefore, each node
constructs a data model to capture the underlying structure of the observed data. In this thesis, the AR(3) model (Brockwell and Davis 1994) is used to regress the sensor data to build the feature model at each sensor node based on the readings taken by them. The model (dis) similarity between each pair of nodes is then found. This feature distance is then used to build the feature regions. A designated node within each region (say, the node with the highest ID) represents all its member nodes and sends only the model coefficients to the CH. Each of the aforementioned steps is elaborated in the following sections.

4.3.1 Construction of AR Model

The time series of a sensor node consists of a trend component that grows slowly over time, a seasonal component with some periodicity and a stationary component. However, in order to reduce the storage and computational requirements of individual sensor nodes, the trend and seasonal components are normally ignored. As discussed in Brockwell and Davis (1994), if the time elapsed since the last reading is relatively short, it is reasonable to neglect those components. Therefore, an AR(3) model is adopted in the thesis.

In an AR (p) model, the time series of an attribute V at any node is modeled as,

$$V_t = \alpha_1 V_{t-1} + \ldots + \alpha_k V_{t-k} + \varepsilon_t$$  \hspace{1cm} (4.1)

where $\alpha_1, \ldots, \alpha_k$ are the auto-regression coefficients and $\varepsilon_t$ is white noise with a zero mean and non-zero variance.
The sensor data from the Intel, Berkeley research lab (http://db.csail.mit.edu/labdata/labdata.html, Dec 2006), which consists of about a month’s worth of temperature readings collected approximately every thirty seconds from 20 sensor nodes\(^1\) are used for the purpose of the study. A sample of time-series data collected from 5 random nodes is shown in the Figure 4.1.

![Time series data of the temperature data observed by the sensor nodes](image)

**Figure 4.1 Time Series Chart of the Sensor Readings of 5 Nodes**

A *learning window* of size \(N=60\) readings is assumed. Each node models the feature (here, temperature) after performing \(N\) readings on that feature. Let \(V_1, \ldots, V_N\) be the temperature values read from each sensor, and \(\overline{V}_1, \ldots, \overline{V}_N\) are calculated as,

\(^1\) Most of the temperature readings of node 5 are missing in the original data set and hence node 5 is omitted in all further calculations.
where \( \eta \) is the mean of \( N \) readings

The model coefficients are then obtained via least-squares regression by minimizing the function \( Q \) (Tulone 2006):

\[
Q(\alpha_1, \alpha_2, \alpha_3) = \sum_{i=4}^{N} \left[ \bar{V}_i - \left( \alpha_1 \bar{V}_{i-1} + \alpha_2 \bar{V}_{i-2} + \alpha_3 \bar{V}_{i-3} \right) \right]^2
\]

The coefficients \( \alpha_1, \alpha_2, \alpha_3 \) can be computed by setting the partial derivatives of the minimum squared error to zero and solving a linear system of the following three equations:

\[
\alpha_1 \sum_{i=4}^{N} \bar{V}^2_{i-1} + \alpha_2 \sum_{i=4}^{N} \bar{V}_{i-1} \bar{V}_{i-2} + \alpha_3 \sum_{i=4}^{N} \bar{V}_{i-1} \bar{V}_{i-3} = \sum_{i=4}^{N} \bar{V}_i \bar{V}_{i-1} \tag{4.4}
\]

\[
\alpha_1 \sum_{i=4}^{N} \bar{V}_{i-1} \bar{V}_{i-2} + \alpha_2 \sum_{i=4}^{N} \bar{V}^2_{i-2} + \alpha_3 \sum_{i=4}^{N} \bar{V}_{i-2} \bar{V}_{i-3} = \sum_{i=4}^{N} \bar{V}_i \bar{V}_{i-2} \tag{4.5}
\]

\[
\alpha_1 \sum_{i=4}^{N} \bar{V}_{i-1} \bar{V}_{i-3} + \alpha_2 \sum_{i=4}^{N} \bar{V}_{i-2} \bar{V}_{i-3} + \alpha_3 \sum_{i=4}^{N} \bar{V}^2_{i-3} = \sum_{i=4}^{N} \bar{V}_i \bar{V}_{i-3} \tag{4.6}
\]

By solving the Equations (4.4), (4.5) and (4.6), the model coefficients \( \alpha_1, \alpha_2 \) and \( \alpha_3 \) are obtained.

A sample set of 60 temperature readings of 20 nodes is given in Table A1.1. Node 2 is taken for illustration and the AR (3) model of sensor node 2 taken during the learning window is obtained as follows: Each of the \( N \) readings of sensor 2 is subtracted from the mean value \( \eta \) of \( N \) readings and the values are substituted in the Equations (4.4), (4.5) and (4.6). The following matrices are obtained:
By solving the above matrices for unknowns using the Gauss-elimination method, the model coefficients obtained are as follows:

\[
\begin{bmatrix}
0.6848 & 0.6693 & 0.6615 \\
0.6693 & 0.6814 & 0.6717 \\
0.6615 & 0.6717 & 0.6906
\end{bmatrix}
\begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\alpha_3
\end{bmatrix}
=
\begin{bmatrix}
0.6826 \\
0.6706 \\
0.6629
\end{bmatrix}
\]

Therefore, the AR model coefficients of sensor node 2 thus obtained are 0.8674, 0.1236 and 0.0089 respectively. In a similar way, the AR models are obtained for all the other sensor nodes.

Generally, AR(p) model fits well for linear data. It is observed that, the sensor network data is typically locally linear, but there are periodic non-linearities that are not well predicted by AR(p) models (Tulone 2006). To solve this problem, every node dynamically updates its model coefficients as necessary; the idea is to detect when the model already learned is no longer a good fit for the data currently being sensed, and dynamically re-learn the model coefficients when such phenomenon occurs.

4.3.2 Distance Calculation

The feature model of sensor node i is denoted by Fi. The feature distance between any two models Fi and Fj is captured by distance d(Fi, Fj), which is measured using the Euclidean distance between their LPC cepstral coefficients (Kalpakis et al 2001).
The cepstrum of an ARIMA time-series can be estimated using the parameters of the AR model for that time-series. The cepstrum defined using the AR coefficients is referred to as the LPC cepstrum, since it is derived through Linear Predictive Coding models for the time series (Furui 1989). The cepstral coefficients for an AR(p) time-series can be derived from the auto-regression coefficients as shown in Equation (4.7). They provide more discriminating power and the ability to separate the time-series generated by different models.

\[
c_n = \begin{cases} 
- \alpha_1 & \text{if } n=1 \\
- \alpha_n - \sum_{m=1}^{n-1} \left(1 - \frac{m}{n}\right) \alpha_m c_{n-m} & \text{if } 1<n\leq p \\
- \sum_{m=1}^{p} \left(1 - \frac{m}{n}\right) \alpha_m c_{n-m} & \text{if } n>p 
\end{cases}
\]  

(4.7)

Here, \(c_n\) refers to the \(n^{th}\) cepstral coefficient and \(p=3\) for an AR(3) model. The cepstral coefficients are used to extract the significant features of time-series and the distance between two time series is defined as the weighted Euclidean distance.

The distance \(d (F_1, F_2)\) is calculated as

\[
d (F_1, F_2) = \sum_{i=1}^{p} W_i \left(\alpha_{i1} - \alpha_{i2}\right)^2
\]

(4.8)

where,

\(\alpha_{i1}\) and \(\alpha_{i2}\) are the \(i^{th}\) AR coefficients of \(F_1\) and \(F_2\)

\(w_i\) (weight for the \(i^{th}\) AR coefficient) = \(c^i / K\)

and \(K = \sum_{i=1}^{p} c^i\) for some constant \(0 < c < 1\)
For example, for the AR model coefficients of the sensor node 2, 
\( \alpha_1 = 0.8674, \alpha_2 = 0.1236 \) and \( \alpha_3 = 0.0089 \), the cepstral coefficients are calculated using the Equation (4.7) as follows:

\[
c_1 = -\alpha_1 = -0.8674
\]

\( c_2 \) is calculated by substituting \( n=2 \) in Equation (4.7)

\[
c_2 = -\alpha_2 - (1-(1/2)) \alpha_1 c_1
\]
\[
= -0.1236 - (0.5)(0.8674)(-0.8674)
\]
\[
= -0.1236 + 0.3762
\]
\[
= 0.2526
\]

\( c_3 \) is calculated by substituting \( n=3 \) in Equation (4.7)

\[
c_3 = -\alpha_3 - (1-(1/3)) \alpha_1 c_2 - (1-(2/3)) \alpha_2 c_1
\]
\[
= -0.0089 - (0.66)(0.8674)(0.2526) - (0.33)(0.1236)(-0.8674)
\]
\[
= -0.118
\]

Hence, the cepstral coefficients of sensor node 2 obtained are 
\(-0.8674, 0.2526 \) and \(-0.118 \).

Then the distance between every node to every other node is calculated using the weighted Euclidean distance using the Equation (4.8). For example, the distance between sensor nodes 1 and 2 is calculated as 0.01. Similarly, the distance is calculated for all the nodes. The distance matrix formed for a subset of 9 nodes is shown in Table 4.1.
Table 4.1 Distance Matrix for a Sample of 9 Nodes

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.01</td>
<td>0.06</td>
<td>0.22</td>
<td>0.01</td>
<td>0.3</td>
<td>0.53</td>
<td>0.01</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
<td>0</td>
<td>0.11</td>
<td>0.15</td>
<td>0.05</td>
<td>0.22</td>
<td>0.72</td>
<td>0.05</td>
<td>0.12</td>
</tr>
<tr>
<td>3</td>
<td>0.06</td>
<td>0.11</td>
<td>0</td>
<td>0.55</td>
<td>0.03</td>
<td>0.66</td>
<td>0.32</td>
<td>0.08</td>
<td>0.04</td>
</tr>
<tr>
<td>4</td>
<td>0.22</td>
<td>0.15</td>
<td>0.55</td>
<td>0</td>
<td>0.32</td>
<td>0.01</td>
<td>1.3</td>
<td>0.28</td>
<td>0.44</td>
</tr>
<tr>
<td>6</td>
<td>0.01</td>
<td>0.05</td>
<td>0.03</td>
<td>0.32</td>
<td>0</td>
<td>0.43</td>
<td>0.35</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>7</td>
<td>0.3</td>
<td>0.22</td>
<td>0.66</td>
<td>0.01</td>
<td>0.43</td>
<td>0</td>
<td>1.29</td>
<td>0.35</td>
<td>0.54</td>
</tr>
<tr>
<td>8</td>
<td>0.53</td>
<td>0.72</td>
<td>0.32</td>
<td>1.3</td>
<td>0.35</td>
<td>1.29</td>
<td>0</td>
<td>0.26</td>
<td>0.16</td>
</tr>
<tr>
<td>9</td>
<td>0.01</td>
<td>0.05</td>
<td>0.08</td>
<td>0.28</td>
<td>0.01</td>
<td>0.35</td>
<td>0.26</td>
<td>0</td>
<td>0.02</td>
</tr>
<tr>
<td>10</td>
<td>0.05</td>
<td>0.12</td>
<td>0.04</td>
<td>0.44</td>
<td>0.02</td>
<td>0.54</td>
<td>0.16</td>
<td>0.02</td>
<td>0</td>
</tr>
</tbody>
</table>

The minimum spanning tree clustering algorithm is then used to partition the nodes into a set of disjoint δ-regions of spatial similarity i.e. all the nodes within a given region are within distance δ from each other. Higher the spatial stability within the cluster, lower will be the number of feature regions.

4.3.3 The Minimum Spanning Tree Clustering Algorithm

A minimum spanning tree of a weighted graph G is the minimum-weight connected acyclic subgraph of G containing all vertices of G (Evans and Edward 1992). The algorithm consists of two phases. In the first phase, the algorithm picks up an arbitrary node as a root and builds the spanning tree on the given communication graph of the network. In the second phase, the tree is partitioned into sub trees such that \( d(F_i, F_j) \leq \delta \). The algorithm that runs at an arbitrary node k is shown in Figure A 2.1 (Meka and Ambuj 2006).
Figure 4.2 shows the initial communication graph considered. Figure 4.3 shows the output of phase 1, i.e., after constructing the minimum spanning tree of the given communication graph and Figure 4.4 shows that the spanning tree clustering algorithm partitions the initial network into 3 feature regions. Region 1 contains nodes measuring temperature values around 19°C, region 2 contains nodes in the range of 17°C, while region 3 is formed only by nodes 16 and 20, both measuring temperature in the range of 16°C.

![Initial Communication Graph Figure 4.2](Image)

![Depth First Search (DFS) Traversal in WSN Figure 4.3](Image)
The first phase of the spanning tree clustering algorithm picks sensor node 1 as a root and builds the spanning tree using Depth First Search (DFS) traversal on the communication graph of the network as shown in the Figure 4.3.

In the second phase, clustering based on feature distance is done as shown in the Figure 4.4. Two nodes are not grouped in the same cluster, if the distance between their features exceeds the global dissimilarity threshold $\delta=2.9$ (chosen empirically). The value of $\delta$ is decided based on how much of demarcation of the feature space is required and is left as a choice of the design. Understandably, a smaller value of $\delta$ would result in more number of feature regions and vice versa.
As discussed in the spanning tree algorithm given in Appendix 2, each node maintains the variable ‘height’. Initially, it is set to zero for all the sensor nodes. The height of each sensor node i, ‘height_i’ is updated to ‘height_i + d(F_i, F_j)’. d(F_i, F_j) is the distance between the features (F) of the sensor nodes ‘i’ and ‘j’.

Height of Node 1 is 0.00
Height of Node 2 is 0.01
Height of Node 3 is 0.12
Height of Node 4 is 0.67
Height of Node 6 is 0.99
Height of Node 7 is 1.42
Height of Node 8 is 2.03
Height of Node 9 is 1.77
Height of Node 10 is 2.19

Height of Node 11 is 2.92

Here, the height of the node 11 exceeds the chosen threshold δ. Hence node 11 does not cluster with nodes 1,...,10. Hence, the nodes 1,2,3,4,6,...10 form region 1. In a similar manner, other regions are formed as follows:

Height of Node 11 is 0
Height of Node 12 is 0.26
Height of Node 13 is 0.4
Height of Node 14 is 0.84
Height of Node 17 is 0.26
Height of Node 18 is 0.04
Here, the height of the node 16 exceeds the threshold δ. Hence the nodes 11, 12, 13, 14, 15, 17, 18, 19 and 20 form region 2.

Nodes 16 and 20 form region 3. Hence, three feature regions are formed as a result of the spanning tree clustering algorithm for the network under consideration.

4.4 THE MODEL UPDATION ALGORITHM

The mean \( \eta \) and the coefficients \( \alpha_1, \alpha_2, \alpha_3 \) uniquely describe the AR model for a given set of learning data \( \{v_1, \ldots, v_N\} \); instead of sending the raw data from the sensor nodes, the model co-efficients are sent to the sink node via the region heads. The CH maintains one AR model per feature region.

In order to keep track of the changes occurring in the phenomenon being sensed, the AR model must be dynamic and accordingly the cluster memberships should change over time. To accomplish this, each sensor periodically monitors its local model to find whether it properly fits the input data. When the model ceases to properly fit the observed sensor data, the node
updates the model and dynamic re clustering takes place according to the updated model.

Each sensor models the feature using its readings in the learning window as explained in section 4.3.1. From the model, the feature regions are formed as found in section 4.3.3. Then during a time frame called the observation window (\( \tau = 300 \) seconds), the current reading of a node is extrapolated from the linear sequence of the previous 3 readings as follows: (Tulone 2006)

\[
P(t) = \eta + \alpha_1 (V_{t-1} - \eta) + \alpha_2 (V_{t-2} - \eta) + \alpha_3 (V_{t-3} - \eta) \tag{4.9}
\]

where, \( P(t) \) is the value predicted for the reading at time \( t \).

For the given input data set, the first three readings are predicted using the following equations:

\[
P_1 = \eta \tag{4.10}
\]

\[
P_2 = \eta + \alpha_1 (V_1 - \eta) \tag{4.11}
\]

\[
P_3 = \eta + \alpha_1 (V_2 - \eta) + \alpha_2 (V_1 - \eta) \tag{4.12}
\]

\( V_1 \) and \( V_2 \) represent the first and second readings measured by a sensor node. The predicted readings \( P_4 \) to \( P_N \) are then obtained by using the Equation (4.9).

Two thresholds are used namely, \( \Omega \) and \( \mu = \gamma b(\tau) \) where \( \gamma \) is the confidence parameter on predicted sensor readings (\( \gamma = 5 \)) and \( b(\tau) \) is the variance of the prediction error in the predicted value.
By making use of the prediction model given in Equation 4.9 to 4.12, each of the readings taken during the observation window is compared against its corresponding predicted value and the prediction error is calculated in order to verify the validity of the model. Then using the aforementioned two thresholds, if the absolute value of the prediction error falls in \([0, \Omega]\), then the model is a good predictor of the data If it falls in \([\Omega, \mu]\) the data is still within the error bound but the model might need to be updated. Finally, if the error in prediction exceeds \(\mu\), then the data is an outlier (Tulone 2006). The model should be updated only when the observed readings diverge consistently from its model i.e., if more than 50% of the readings in the observation window cause outliers, then the model coefficients are re-calculated and the new model is communicated to the CH via the respective region head.

\(b(\tau)\) is calculated from the prediction errors as follows: The prediction error \(e\) of the readings taken, \(e_t = P_t - \overline{V}_t\), for \(t = 1, \ldots, N\) and using \(\overline{e} = (e_1 + e_2 + \ldots + e_N) / N\), the variance of the white noise is,

\[
b(\tau) = \left( \sum_{t=1}^{N} (e_t - \overline{e})^2 / (N - 1) \right)^{1/2} \quad (4.13)
\]

As an example, sensor node 2’s readings \(\{V_1, \ldots, V_{60}\}\) taken during the learning phase are considered for illustration. The \(\overline{V}_t\) readings are used such that \(\overline{V}_t = V_t - \eta_t\) for \(t = 1\) to 60. The model coefficients of node 2 calculated during the learning window are used to predict the readings as explained in section 4.4 using the Equations (4.9) - (4.12). Next the error value \(e_t\) is calculated.
The mean value of the readings \( \eta \) of node 2 is 19.336. Using the AR model coefficients of node 2 already calculated as,

\[
\begin{align*}
\alpha_1 &= 0.8674 \\
\alpha_2 &= 0.1236 \\
\alpha_3 &= 0.0089
\end{align*}
\]

the predicted values of the node 2’s readings are obtained as follows:

\[
\begin{align*}
P_1 &= \eta = 19.336 \\
P_2 &= \eta + \alpha_1(V_1 - \eta) \\
&= 19.336 + 0.8674(0.231) \\
&= 19.536 \\
P_3 &= \eta + \alpha_1(V_2 - \eta) + \alpha_2(V_1 - \eta) \\
&= 19.336 + 0.8674(0.2016) + 0.1236(0.231) \\
&= 19.539
\end{align*}
\]

Similarly, the values for \( P_4 \) to \( P_N \) are calculated using the Equation (4.9). Subsequently, the error in prediction is calculated as below:

For the first reading of node 2 under the given learning window,

\[
\begin{align*}
e_1 &= P_1 - \bar{V}_1 \\
&= 19.336 - 0.231 \\
&= 19.105
\end{align*}
\]

After obtaining the error in prediction for all the readings taken by node 2 as shown in Table A1.2, the mean error value is calculated as follows,
\[ e = 19.105 + \ldots + 19.327 \]

\[
\frac{\text{------------------------}}{60} = 19.3398
\]

The variance in error values is then calculated as,

\[
b(\Gamma) = \left( \frac{(19.105 - 19.3398)^2 + \ldots + (19.327 - 19.3398)^2}{60-1} \right)^{1/2}
\]

\[
= 0.2395
\]

i.e., the variance is 0.2395 and therefore \( \mu = 1.2 \) for \( \gamma = 5 \).

With \( \Omega = 0.5 \) (chosen empirically) and \( \mu = 1.2 \), the model updation algorithm is executed with the input readings taken during a given observation window. A subset of the readings shown in the Table 4.2 of the sensor nodes 11, 12, 13, 14 and 15 illustrates the re-clustering process.

Table 4.2 Partial Data Set of Sensor Nodes used for Model Update Algorithm

<table>
<thead>
<tr>
<th>Node 11</th>
<th>Node 12</th>
<th>Node 13</th>
<th>Node 14</th>
<th>Node 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.84</td>
<td>18.5376</td>
<td>16.5484</td>
<td>18.3024</td>
<td>16.5388</td>
</tr>
<tr>
<td>16.6484</td>
<td>18.8494</td>
<td>17.0023</td>
<td>18.175</td>
<td>16.5682</td>
</tr>
</tbody>
</table>

For the given set of readings, according to the model updation algorithm discussed above, at nodes 11, 13 and 15, the outlier count exceeds 50% of the observation window (size of the observation window is considered as 10); now these sensor nodes relearn their model and notify the new model.
coefficients to their CH. If the observed deviation is large enough to violate the condition for membership in a given region (distance exceeds $\delta$), then re-clustering takes place according to the new model parameters recomputed. The network after re-clustering is shown in Figure 4.5.

**Region 1**

![Region 1 diagram]

**Region 2**

![Region 2 diagram]

**Region 3**

![Region 3 diagram]

**Figure 4.5 The Network after Re clustering**

As seen in Figure 4.5, the nodes 11, 13 and 15 now join region 3 as a result of the changes occurring in the readings measured by these nodes.

The complete sequence of events taking place as discussed in the sections 4.3-4.5 is illustrated using a flow diagram given in Figure 4.6.
Figure 4.6 Flow Diagram of the Complete Framework

Start

For every node in the network

Calculate AR (3) coefficients on Learning Window readings

From every node to every other node

Form Distance Matrix

Cluster regions and select the region head

Notify AR (3) coefficients to region head

For every node in the region

For every reading in observation window

Predict the readings using prediction model

Calculate the prediction error

Indicates ‘No change’ to region head

No of outliers >50% of Observation Window size

Relearn model and notify updated parameters

Stop

Size of Learning Window = 60 readings
Size of Observation Window = 10 readings
\( \Omega = 0.5 \)
\( \mu = 1.2 \)

Size of Learning Window

Size of Observation Window
4.5 SUMMARY

The local analysis of the sensed values not only leverages the processing potential of individual sensor nodes but also helps to reduce the communication overhead. This chapter discussed about how the temporal-spatial correlation present in the environmental data can be made use of in reducing the communication overhead involved in continuous data gathering applications. Each sensor node builds a model of the feature being observed and feature regions are constructed based on the model similarity among the nodes. This necessitates sending only the model coefficients to the CH from each of the feature regions, thus immensely saving the communication overhead. The CH then uses these model coefficients to predict the actual sensor readings and subsequently answer queries from the base station. Since feature regions are clearly demarcated, any topographic query will be appropriately routed to the relevant feature region.