INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2007; **20**:829–856
Published online 24 October 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac.847

Using polynomial regression for data representation in wireless sensor networks

Torsha Banerjee[‡], Kaushik R. Chowdhury[§] and Dharma P. Agrawal*[,†]

OBR Center for Distributed and Mobile Computing, ECECS, University of Cincinnati, Cincinnati, OH 45221-0030, U.S.A.

SUMMARY

Unlike conventional sensor networks, wireless sensors are limited in power, have much smaller memory buffers, and possess relatively slower processing speeds. These characteristics necessitate minimum transfer and storage of information in order to prolong the network lifetime. In this paper, we exploit the spatio-temporal nature of sensor data to approximate the current values of the sensors based on readings obtained from neighbouring sensors and itself. We propose a tree based polynomial regression algorithm (TREG), that addresses the problem of data compression in wireless sensor networks. Instead of aggregated data, only the coefficients computed by the regression function, TREG are passed to achieve the following goals: (i) the sink can get attribute values in the regions devoid of sensor nodes, and (ii) readings over any portion of the region can be obtained at one time by querying the root of the tree. As the size of the data packet from each tree node to its parent remains constant, the proposed scheme scales very well with growing network density or increased coverage area. Since physical attributes exhibit a gradual change over time, we propose an iterative scheme, UPDATE_COEFF, which obviates the need to perform the regression function repeatedly and uses approximations based on previous readings. Extensive simulations are performed on real world data to demonstrate the effectiveness of the aggregation algorithm, TREG. Results reveal that for a network density of 0.0025, a complete binary tree of depth 4 could provide the absolute error to be less than 6%. A data compression ratio of about 0.02 is achieved using our proposed algorithm, which is almost independent of the tree depth. In addition, our proposed updating scheme makes the aggregation process faster while maintaining the desired error bounds. Copyright © 2006 John Wiley & Sons, Ltd.

Received 20 January 2006; Revised 21 June 2006; Accepted 30 June 2006

KEY WORDS: aggregation; attribute-based trees; dummying; polynomial regression; sensor network; spatio-temporal

*Correspondence to: Dharma P. Agrawal, Department of Electrical and Computer Engineering and Computer Science, University of Cincinnati, Cincinnati, OH 45221-0030, U.S.A.
[†]E-mail: dpa@ececs.uc.edu
[‡]E-mail: banerjta@ececs.uc.edu
[§]E-mail: kaushir@ececs.uc.edu

Contract/grant sponsor: Ohio Board of Regents' Doctoral enhancement Funds

1. INTRODUCTION

Recently, many practical applications like environmental monitoring, military installations and scientific research have been proposed [1] using wireless sensor networks (WSNs). Such applications necessitate transfer of a huge amount of relevant data from one point of the network to another. This forces the use of a fast and robust data aggregation protocol that could compress data without any substantial loss in accuracy, thereby facilitating quick data storage and retrieval. Compression and aggregation directly translates to energy savings, and this is a crucial issue in WSNs. Therefore, data aggregation may be optimized in terms of latency and energy by judicious choice of the number of aggregation points and their locations [2], reduction in total number nodes used for transmitting sensed data to the sink as in LEACH [3], formation of special boundaries like Voronoi tessellations [4], amongst others. Sensed parameters, like almost any other physical attribute, exhibit a gradual and continuous variation over 2-D Euclidean space. Our scheme is based on the fact that there is a correlation between attribute values and location as presented in Reference [5], and hence between sensors in close proximity [6]. Furthermore, a close analysis of sensor data in naturally occurring physical phenomenon exhibits spatial correlation. Tree-based polynomial regression (TREG) [7] leverages this phenomenon by aggregating correlated attribute values and eliminating underlying redundancy. Our scheme first creates multiple attribute-based binary trees (the key aggregation data structure of TREG also called query trees or QTs) and then based on the network sensor density derives a probabilistic bound with which a node joins the tree. After the tree construction phase, sensors which are non-tree nodes or NT nodes) report the sensed values, $f(x, y)$ (a function of the location (x, y)) to the tree nodes closest to them. Each tree sensor node (solely does aggregation and not sensing) then performs polynomial regression on the reported values to compute the coefficients $\beta_0, ..., \beta_8$ of the regression polynomial $\beta_0 + \beta_1 y + \beta_2 y^2 + \beta_3 x + \beta_4 xy + \beta_5 xy^2 + \beta_6 x^2 + \beta_7 x^2 y + \beta_8 x^2 y^2$, which is passed on to the higher level in place of raw data. Thus, nodes at each level use the coefficients of their children to improve the approximation function and this procedure is repeated till the root is reached. The sink (the base station which receives queries from end users and itself queries the root of QT) can finally have access to an approximation, $f(x, y)$ of the sensed attribute at any point in the region spanned by the tree. The root is the topmost node of QT which receives the final polynomial and gets answers to queries sent by the sink. The attribute values at any point (x, y) can be obtained by substituting the values of x and y in the final polynomial at the root. For sparse networks, QT might not be complete, so the parent needs to approximate the information for the missing child based on the data available from the other neighbouring child. On the other hand, if no child is present, the parent uses its own data to approximate information for its missing children. This method of substituting attributes which we formally call as DUMMYREG [8] increases the accuracy of the overall approximation process by including readings from regions devoid of actual sensor nodes. Apart from spatial correlation being an important characteristic of sensors, sensor readings are also observed to be temporally correlated. Thus, for successive rounds of aggregation, instead of performing regression at each tree node periodically, our proposed algorithm UPDATE_COEFF simply modifies previously computed coefficients (which can be stored in a table in the sensor's memory and retrieved through efficient table look-up) at each node. The modified coefficients are then sent up the tree using TREG, assuming a smooth variation from their previous values. In TREG, a fixed sized data packet is sent up the tree by each node thus keeping the bandwidth usage fixed. Again, as

DOI: 10.1002/dac

each packet consists of only coefficients and two pairs of $x-y$ co-ordinates, the overall bandwidth efficiency is also improved considerably. The rest of this paper is organized as follows: Section 2 lists the preliminaries and discusses existing work. Section 3 describes our proposed scheme, TREG in details. This section formally introduces our polynomial regression algorithm for achieving data compression, TREG. In Section 4, we discuss a way to optimize TREG for varying topologies (when QT is not complete). In Section 5, TREG is further optimized by introducing a new scheme. Instead of performing regression every time the attribute values change, the idea of this scheme is to update the coefficients of the previous period thus decreasing the overall computation time. Section 6 discusses the construction of QT in details. We first propose an algorithm which determines how to select the root of QT followed by discussion about the formation of each QT similar to Reference [9]. Expressions for optimal depth of QT are derived in this section. Simulation results are presented in Section 7. Finally, Section 8 concludes this paper.

2. RELATED WORK

In a typical Greedy aggregation-tree approach [2], a shortest path is built between the first source to the sink and subsequent sources are connected to the closest nodes of the existing tree by creating incremental least energy paths. This scheme is useful in providing savings for proactive systems (systems which require periodic updates). However, for queries with attribute in a given range, a fairly expensive network-wide flooding [10] is used. Also, sink is assumed to be immobile, as gradients remain unchanged during the operation of the scheme. In LEACH [3], a set of nodes is selected randomly as cluster heads (CHs) and each node joins a cluster depending on the communication energy between the node and the CH. Also, to preserve energy, the role of CH needs to be changed. However, a major limitation of this scheme is that the CHs themselves may run out of energy to transfer data to the sink as they have to do extra work to serve as CHs and the sink is assumed to be situated far away from them. In TREG, the continuous use of the tree nodes amounts to passing a set of nine coefficient values at periodic intervals and this is the overall energy cost of a sensor reporting its own reading (please recall that the tree node does not have to report its own reading to anyone). Thus no additional energy usage occurs in the tree structure as compared to (maybe a fractional more) the other ground level sensing nodes i.e. NT nodes.

In Reference [4], Voronoi clusters are formed by associating each node with a particular sink. A border node is a node which belongs to more than one such cluster. If a sink generates a query involving more than one cluster, the corresponding border node needs to route the query every time from the internal nodes to the sink and is termed as a bottleneck node.

The density of the network [11] is observed to have a dominant effect on the distortion at the sink as transmitting power level is adjusted such that adjacent sensors can interchange their values. As the density increases in multi-hop sensor networks with adjustable transmission power, the number of hops between any two arbitrary nodes also increases proportionately. This increases the delay in the network and accordingly decreases the temporal correlation of sensors. Increase in node density also increases the number of neighbours, thus enhancing the spatial correlation among them. Using node density as the determining factor, accuracy of estimation and energy efficiency in retrieval of sensor data at the sink is analysed in Reference [11]. The goal of the scheme is to derive an optimal value of N, the number of nodes in the

DOI: 10.1002/dac

network that minimizes the spatio-temporal distortion. For a two-dimensional square grid network of size $L \times L$, the optimal value of N decreases when the values in the region become closely correlated. A polynomial equation in N can be derived based on the distortion $D(N)$. This scheme primarily focuses on energy saving techniques by exploiting the spatio-temporal correlation of sensor nodes. We propose a scheme for updating the reading stored previously at a sensor with an assumption that consecutive sensor readings are similar in nature [12]. This saves time in computing new coefficients, resulting in lower delays, and sending coefficients in place of raw sensor data saves bandwidth and hence energy consumption in the process.

In Reference [10], a node is elected as the representative node for sending the snapshot of a sensed region to the sink. Though this scheme reduces the overall number of nodes required to satisfy a query request, it nevertheless involves election of the representative node that needs to be determined at definite intervals of time (making it strictly proactive). TREG, unlike Reference [10], avoids this extra work by selecting a random node location as the representative of the sensed region in a reactive manner, i.e. the system adopts according to the unpredictability of environmental model.

TREG is similar to Reference [14] where a kernel function fits the measurements taken by a sensor at different intervals of time, the readings being temporally correlated. Again, sensors being spatially correlated, a node also approximates its neighbours' readings with its own. However, every pair of neighbouring nodes need to exchange message which consists of a square matrix and a vector whose size depends on the number of variables shared by the two nodes. On the other hand, in our scheme, each node sends only a constant number of coefficients along with their co-ordinates to it parent. Since the leaf nodes fits a polynomial which consists of only readings from the sensing (NT) nodes spanned by it, a linear equation is adequate for the lowest level of QT, thus requiring only four coefficients in place of nine. As the compression method continues up the tree, each non-leaf node needs to incorporate readings from the NT nodes spanned by it as well as the regenerated measurements obtained from each of its child nodes. A non-linear (quadratic is sufficient) equation is found to capture these increased number of readings accurately conforming to Reference [13]. TREG utilizes Mathematica (explained in Simulations results) for performing regression which gives a quadratic polynomial with nine coefficients making the number of coefficients in our approximation scheme also nine. For maintaining uniformity and simplicity of simulation, we have assumed that each tree node sends nine coefficients to its parent irrespective of its level in the tree. In Reference [14], each node can only answer queries belonging to the particular region covered by its kernel which again depends on the size of the kernel. In our scheme, the root node keeps the final set of coefficients which spans the entire network under it, thereby enabling it to compute values at any point in the network as long as there is a smooth variation in the data distribution.

Similar to the contour maps generated in our scheme, Reference [15] also builds contour maps to depict the nature of spatial correlation in sensors. In this paper, a sensing node suppresses the attribute sensed, if its neighbour also senses the same attribute with a value similar to it within a threshold δ. For this, a node needs to overhear its neighbour's reading and then back off accordingly. In our scheme, sensing of attributes is solely done by NT nodes and data filtering is taken care of by tree nodes. Unlike discontinuities produced by local decisions in Reference [15], our scheme reproduces a contour which is mostly continuous. Finally, to compute the maximum over the readings, the aggregation process needs to be modified at the starting point in Reference [15] so that nodes with less than its neighbour's value suppress their readings. In our

DOI: 10.1002/dac

scheme, the maximum can be computed after the proposed data aggregation process is complete, simply by differentiating the final polynomial at the root.

3. OUR PROPOSED REGRESSION-BASED SCHEME

Our scheme is based on the periodic per-hop timing model [16] of data aggregation along each sub-tree. In a periodic per-hop category timing model, a node sends the aggregated packet as soon as it hears from all its children. However, at the root, only a final message is sent containing the coefficients, thus conforming to a simple periodic timing model. Each node in the network has a unique node ID and has location information through triangulation and other location services [17]. According to Reference [18] several schemes on localization exist in literature and '*no single algorithm performs best; which algorithm is to be preferred depends on the conditions (range errors, connectivity, anchor fraction, etc.)*'. The reader is therefore encouraged to refer to these cited works to gain further knowledge about location estimation in WSNs.

3.1. Proposed regression-based function approximation scheme

Each node of QT stores the attribute value sent by each of the nearest NT sensor nodes. These NT nodes report their data to the tree node closest to them, for storage of the current attribute reading. Recall that NT nodes only sense attributes while the QT is for storage only. As the measured attribute varies with respect to space in a continuous manner, the values stored at the QT node can be considered as function values having two inputs, x and y. Since, attribute (say z) in a spatially correlated region is a function of sensor location which can be denoted by the (x, y) co-ordinates, therefore z is dependent on both the independent variables, x and y, i.e., $z=f(x, y)$. Thus, $z_1 = ((f_1(x_1, y_1)), x_1, y_1)$ is one such attribute tuple where z_1 is the attribute value sensed by a node at location (x_1, y_1). Node i of a QT creates a function approximation $f_i(x, y)$ from the data reported to it by its nearest NT nodes. Following the approach given in Reference [13], we find by incrementing the degree of the polynomial, that a polynomial equation of second degree (quadratic) is sufficient to capture the non-linearity of the measured data set accurately. Increasing it further does not improve the accuracy (percentage error of 6% is reduced to 5.9%) as much as it increases the overhead in sending the increased number of coefficients. Therefore, TREG follows a quadratic polynomial regression approach. It is to be noted here, that higher is the degree of correlation of the data set easier it is to capture it with lower degree polynomial.

Using multivariate polynomial regression [19], a polynomial equation is generated with three input variables ($z=f(x, y)$, x, y) for all the data points in one particular node of QT. A general multilinear regression model can be given as follows [19]:

$$z = f(x_1, x_2, \ldots, x_m) = a_0 + \sum_{k=1}^{m} a_k x_k \tag{1}$$

where $x_1, x_2, ..., x_m$ are the independent variables called predictors of the model and z is the dependent variable. The observations are sampled and the observed values of the vector variable z are used at the particular levels of x_k to estimate a. z is the n-element vector of sample values and $\mathbf{a}$ is the $(m+1) \times 1$ vector estimate of a.

DOI: 10.1002/dac

Applying least-square criterion, the squared error needs to be minimized, i.e.

$$F(\mathbf{a}) = (X\mathbf{a} - \mathbf{z})^{\mathrm{T}}(X\mathbf{a} - \mathbf{z}) \tag{2}$$

where

$$X = \begin{pmatrix} 1 & x_{11} & x_{21} & \cdot\cdot & x_{m1} \\ 1 & x_{12} & x_{22} & \cdot\cdot & x_{m2} \\ : & : & : & : & : \\ 1 & x_{1n} & x_{2n} & \cdot\cdot & x_{mn} \end{pmatrix} \tag{3}$$

$$\mathbf{z} = \begin{pmatrix} z_1 \\ : \\ : \\ z_n \end{pmatrix} K \tag{4}$$

$$a = \begin{pmatrix} a_0 \\ a_1 \\ : \\ a_m \end{pmatrix} K \tag{5}$$

Necessary condition for a minimum is that the partial differentiation of $F(\mathbf{a})$ w.r.t $\mathbf{a}$ is zero.

$$\text{i.e. } \nabla_{\mathbf{a}} F(\mathbf{a}) = \nabla_{\mathbf{a}}(X\mathbf{a} - \mathbf{z})^{\mathrm{T}}(X\mathbf{a} - \mathbf{z}) = \mathbf{0}$$

Again,

$$\begin{aligned}(X\mathbf{a} - \mathbf{z})^{\mathrm{T}}(X\mathbf{a} - \mathbf{z}) &= (\nabla_{\mathbf{a}}(X\mathbf{a} - \mathbf{z}))^{\mathrm{T}}(X\mathbf{a} - \mathbf{z}) + (\nabla_{\mathbf{a}}(X\mathbf{a} - \mathbf{z}))^{\mathrm{T}}(X\mathbf{a} - \mathbf{z}) \\ &= 2X^{\mathrm{T}}((X\mathbf{a} - \mathbf{z})) \\ &= 2X^{\mathrm{T}}X\mathbf{a} - 2X^{\mathrm{T}}\mathbf{z} = 0.\end{aligned}$$

i.e. equated to 0.

The normal equation obtained from above is

$$X^{\mathrm{T}}X\mathbf{a} = X^{\mathrm{T}}\mathbf{z} \tag{6}$$

The system has a solution if $X^{\mathrm{T}}X$ is not singular, i.e. it has an inverse. Therefore, multiplying both sides of Equation (6) by $(X^{\mathrm{T}}X)^{-1}$, we get $\mathbf{a} = (X^{\mathrm{T}}X)^{-1}\ X^{\mathrm{T}}\mathbf{z}$ where $(X^{\mathrm{T}}X)^{-1}\ X^{\mathrm{T}}$, called the (Moore–Penrose-) pseudoinverse of the matrix X is a generalization of the inverse X^{-1}. Using polynomial regression for our model, we get the following equations analogous to Equations

DOI: 10.1002/dac

(3)–(5). In our case, x, y, are the independent variables

$$X = \begin{pmatrix} 1 & y_1 & y_1^2 & x_1 & x_1y_1 & x_1y_1^2 & x_1^2 & x_1^2y_1 & x_1^2y_1^2 \\ 1 & y_2 & y_2^2 & x_2 & x_2y_2 & x_2y_2^2 & x_2^2 & x_2^2y_2 & x_2^2y_2^2 \\ : & : & : & : & : & : & : & : & : \\ 1 & y_n & y_n^2 & x_n & x_ny_n & x_ny_n^2 & x_n^2 & x_n^2y_n & x_n^2y_n^2 \end{pmatrix}, \; z = \begin{pmatrix} z_1 \\ z_2 \\ : \\ z_n \end{pmatrix} \text{ and } \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ : \\ \beta_m \end{pmatrix} \quad (7)$$

For estimating β, a unique inverse of X should exist, i.e., $X^{\mathrm{T}}X$ must be of full rank $m+1$ [20], given that $\boldsymbol{\beta}$ is a $(m+1)\times 1$ vector. In other words, $n \gg m+1$ and no column of X can be expressed as weighted linear combination of any set of other columns.

Here, n is the number sensor locations whose readings are fitted to a polynomial. For each tree node performing regression, $n = n_s$

$$f(x,y) = \beta_0 + \beta_1 y + \beta_2 y^2 + \beta_3 x + \beta_4 xy + \beta_5 xy^2 + \beta_6 x^2 + \beta_7 x^2 y + \beta_8 x^2 y^2 \quad (8)$$

Equation (8) can also be written as

$$f(x,y) = \sum_{i=0,\, j=1}^{i=8,\, j=9} \beta_i.\phi_j \quad (9)$$

where $\phi_1 = 1, \ldots . \phi_9 = x^2.y^2$

Again, each tree node creates the matrix, A_m (from the readings reported by the n NT nodes) for calculating the polynomial. A_m is a square matrix of size 9×9.

The (i–j)th element of

$$Am = \sum_{l=1}^{n_s} \phi_i(x_l, y_l).\phi_j(x_l, y_l) \quad (10)$$

where $i = j$.

At any point of time, each tree node has the tuples, $(x_1, y_1, z_1) \ldots (x_{ns}, y_{ns}, z_{ns})$. Each tree node creates the matrix, F with these tuples. F is a matrix of size, 9×9 where the ith element,

$$F_i = \sum_{l=1}^{n_s} z_l.\phi_i(x_l, y_l) \text{ where } \phi_1 = 1 \ldots \phi_9 = x^2.y^2 \quad (11)$$

At each tree node, β_0, β_1,...β_8 are calculated from Equation (11):

$$\beta = A_m^{-1} \times F \quad (12)$$

Equations (7), (10)–(12) together constitutes the method of Gaussian elimination [21] for solving equation and computing inverse of a matrix.

When $\boldsymbol{\beta}$, obtained from Equation (12), is used with a given location (x, y), we solve $z = f(x, y)$ to retrieve the attribute value at a node location (x, y). Note that $f(x, y)$ is the final function available at the root.

Each tree node uses Equation (12) to generate the coefficients and sends this set to its parent. Nodes at each level regenerate values of the sensed attribute by using these coefficients obtained from their children. These data values are then combined with a node's own reported readings to calculate the new set of coefficients that will be passed to the next higher level. In this process, it is important to identify the region over which the data values are generated as they directly affect the accuracy of the approximation.

DOI: 10.1002/dac

We identify this region as the area bounded by the co-ordinates $\{x_{\min}, y_{\min}, x_{\max}, y_{\max}\}$ where the minimum and maximum are taken over all the sensing nodes in the sub-tree under the current parent that report to tree nodes. As an example, consider A_t in Figure 1, as the current aggregation node. The shaded area represents the region in which data values will be re-generated by A_t. This region is bounded by the minimum and maximum co-ordinates of the sensing nodes (diamond-shaped) reporting to the sub-tree under A_t. A_t gets the boundary co-ordinates of this region from its children.

Based on the function approximation process, we have proposed the algorithm, TREG stated in Figure 2. Inputs to the algorithm are the depth and the number of sensing nodes reporting to each tree node. In Figure 2, let $\mathbf{B}(m)$ be the vector representing the coefficient values (β_0, β_1,...β_8) that is passed from a child node m to its parent using our function send(). Let the set A_i represent all nodes of a tree at level i. Hence $|A_i| = 2^i$, where $i = 0, 1, \ldots, p$. Each node m knows

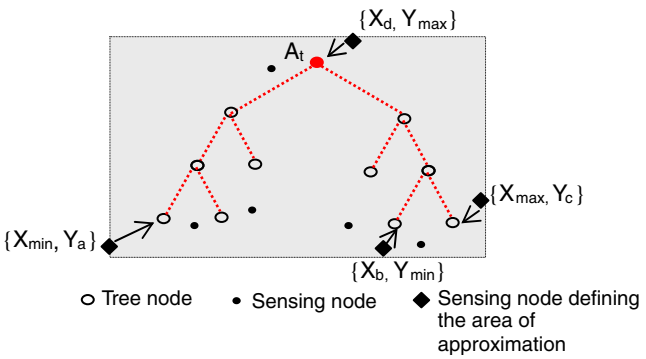

Figure 1. Illustration of how a node calculates the boundary of the region for data regeneration.

PSEUDOCODE *TREG* **(p)**

```
begin
For each level, i=d to 0
∀n, n∈ A_i,
1.Counter =0
2.A_i = A_i −{n}
3.C_n = n.gather()
∀ m, m.parent = n
        4. C_n = C_n U regenerate (B(m))
         5.Counter = Counter +1
           If Counter =2
           6.    B(n) is calculated from C_n
           endif
  If n = root,
  end
  else n.send(B(n), n.parent)
       endif
endfor
end
```

Figure 2. Tree regression algorithm, TREG.

DOI: 10.1002/dac

its parent represented as *m* parent. The function regenerate() uses the boundary values $\{x_{\min}, y_{\min}, x_{\max}, y_{\max}\}$ calculated earlier to identify the region where the data is reconstructed from the received coefficients. We also introduce the function gather() that obtains all the data points from the NT nodes that sense the attribute around the closest tree node. The set C contains all the data points obtained from the neighbours of a tree node and regenerated from its children.

Assuming an arbitrary node k to be the parent of nodes i and j, each of nodes i and j uses Equation (12) to generate the coefficient tuple (β_{i0}, $\ldots\beta_{i8}$) and (β_{j0}, ,$\ldots\beta_{j8}$), respectively and sends this set to node k. Node k now generates two sets of random (x, y) locations and calculates the corresponding values of the sensed attribute at each such location by using the coefficients sent by nodes i and j, respectively. These two data sets are then appended with k's own reported readings to calculate the new set of coefficients that will be passed to k's parent at the next higher level. This process is continued until the root node is reached which will have the final set of coefficients to be used by the sink. Through the construction of the attribute based trees and the aggregation process described above, our scheme can answer location based SQL type query like 'SELECT *temperature* FROM *sensors* WHERE *location* $=(x, y)$' or 'Give maximum acoustic data in the target area ($10 \leqslant x \leqslant 40, 10 \leqslant y \leqslant 40$) every 3 s.' In the latter case, a set of (x, y) co-ordinates are generated in the specified range and the sink can calculate the maximum attribute value by first knowing the attribute values at each of the locations in the set. When a sink needs to know the attribute value at a particular location (x, y), it sends the query to the root. The query is first propagated down the QT to reach the leaf nodes at the last level. The aggregation model follows a bottom–up approach. Reporting data through the aggregation process should be much more efficient in terms of bandwidth and latency, than sending individual data bits corresponding to a specific geographical co-ordinate. To prove our claim, a parameter called compression ratio is defined as the number of bytes transmitted in QT after compression to the original number of bytes transmitted in QT. Compression ratio (based on a round of data aggregation) is calculated as follows: Assume that QT is a complete binary tree of depth p of 2^p leaf nodes. Each attribute packet of size s_i bytes contains the attribute reading (s_c) and the co-ordinates of the location ($s_x + s_y$) where the reading is taken. The number of bytes input to each leaf node is only this data of size $n_s \times s_i$, since n_s is the total number of sensing nodes reporting to each tree node.

Therefore, $T_l = n_s \times s_i \times 2^p$ where T_l the total number of bytes input to the leaf nodes.

Apart from the attribute readings from the n_s sensing nodes, each non-leaf node gets as input from its two children, the coefficients and the x–y boundaries of the area to be regenerated. Thus, $T_{nl} = (n_s \times s_i + 2 \times (s_x + s_y + s_c)) \times (2^{p+1} - 1 - 2^p)$, where T_{nl} stands for the total number of bytes input to the non-leaf nodes. The output packet (from a tree node) of size $(s_x + s_y + s_c)$ bytes contains the coefficients and the x–y range. The total number of bytes output from all the nodes

$$T_0 = (s_x + s_y + s_c) \times (2^{p+1} - 1) \tag{13}$$

Compression ratio (output: input size),

$$CR = \sum_{\text{alledges}} \frac{\text{no. of bytes transmitte } d \text{ on an edge after compression}}{\text{no. of input bytes on an edge}}$$

$$= \frac{T_o}{T_l + T_{nl}} = \frac{(s_x + s_y + s_c) \times t}{(n_s \times s_i \times t + 2 \times (s_x + s_y + s_c)(t_l - 1))} \tag{14}$$

where t is the total number of tree nodes and t_l is the number of leaf nodes.

DOI: 10.1002/dac

4. OPTIMIZATION OF TREG FOR SPARSE TOPOLOGY

In the description of the TREG algorithm presented in Section 3, we have assumed the QT to be complete. In a sparse network, however, nodes may be out of range of each other after random deployment so that the QT formed might not be fully balanced. This section provides a modification of TREG, when the QT is not complete. Algorithm, DUMMYREG (shown in Figure 3) exploits the idea of spatial correlation of attribute values to create readings at locations devoid of actual aggregating nodes, which we define as 'dummying' [8]. It can be called from TREG by adding an extra line of code, '*Call DUMMYREG (index of the missing node)*' in Figure 1 before step 2(d). This ensures that that the readings from missing nodes are still incorporated in the polynomial so as not to compromise with the accuracy of the overall approximation process. For any arbitrary parent node, k of QT, if any of its child nodes is absent, k needs to dummy (or virtualizes) the readings of its missing child. The parent node k

PSEUDOCODE DUMMYR*EG* (*i*)

begin

for each of the non–leaf nodes k of the tree,

CaseI: *k* has only node i as its child and *i* even (*i* is left child), it computes random x-y points for *i+1* where (*xmin, ymin*) and (*xmax, ymax*) are the coordinates of the leftmost and down most node and rightmost and top most node respectively reporting to node *i+1* It sets **B**(i+1) to **B**(i) to regenerate readings for node *i+1*.

```
xmin[i+1]=xmax[i];
xmax[i+1]=xmax[k]+w;
ymin[i+1]=ymin[i];
ymax[i+1]=ymax[i]
```

Case II: node only i+1 is present and *i+1* is odd (*i+1* is the right child), it computes random x-y points for i where (*xmin, ymin*) and (*xmax, ymax*) are the coordinates of the leftmost and down most node and rightmost and top most node respectively reporting to the node *i*. It sets **B**(i) to **B**(i+1) to regenerate readings for *i*.

```
xmin[i]=xmin[k]-w;
xmax[i]=xmin[i+1];
ymin[i]=ymin[i+1];
ymax[i]=ymax[i+1],
```

Case III: k has no children; k performs regression on its reported values and generates coefficients, **B**(k). It sets **B**(i) and **B**(i+1) to **B**(k) to regenerate readings for each of its non-existent children.

```
cc=ymax[k]-ymin[k];
xmin[i]=xmin[k];
xmax[i]=Ceiling[(xmin[k]+xmax[k])/2];
ymin[i]=ymin[k]-cc;
ymax[i]=ymin[k];
xmin[i+1]=Ceiling[(xmin[k]+xmax[k])/2];
xmax[i+1]=xmax[k]+w;
ymin[i+1]=ymin[k]-cc;
ymax[i+1]=ymin[k];
```

endfor

Using **B**(i) and **B**(i+1), new attribute values are calculated and added to the readings sent by nearest NT nodes to node"k"'. Node k then calls the *TREG* to calculate B(k) and passes it to its parent.

endwhile

level=sum

endwhile

end

Figure 3. Algorithm for dummying nodes for an unbalanced QT, *DUMMYREG*.

DOI: 10.1002/dac

generates readings by generating random node locations in the virtual area spanned by each of its virtual children, i or/and $i+1$. In cases I and II (see Figure 3), coefficients of the real child (the child node present) are used to regenerate attribute readings for the virtual child (the child node absent). In case III (see Figure 3), since the parent node has no children, therefore, it uses its own coefficients to regenerate readings for its two virtual children. This method of dummying attribute readings increases the accuracy of the overall compression process by including readings from regions devoid of actual sensor nodes. Without any dummying, i.e. without the inclusion of attribute values from non-existent nodes in the overall approximation process, error incurred is the largest. This is evident from the fact that readings that would have otherwise been reported from the region spanned by the non-existent nodes are not considered in the compression process. With partial dummying, a parent node follows case III of DUMMYREG to regenerate attribute values of both of its non-existing children. But, regeneration is not done for the case when one child is present. For the full dummying case, the error incurred is minimum as attribute values are included from the entire region irrespective of whether a node is actually present or not. The length of each square area spanned by a child is assumed to be a constant w units. When node k does not have a child, the sensing region spanned by its missing child can be approximated to lay w units away from k's sensing region and accordingly its $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$ are adjusted as in Figure 3. The child's y co-ordinate is assumed to be w units below that of the parent. As in Figure 4 (analogous to Figure 1), consider A' as the current aggregating node. The shaded area represents the region in which data values will be regenerated by A'. This region is bounded by the minimum and maximum co-ordinates of the sensing nodes (coloured gray) reporting to the sub-tree under A'. A' gets the boundary co-ordinates of this region from its children. Here, nodes B and D are not present (unlike Figure 2) and therefore they are the dummy nodes. However, the area bounded by them is computed from their neighbours' $\{x_{\min}, y_{\min}, x_{\max}, y_{\max}\}$ ranges using DUMMYREG algorithm. Thus, A' still covers the entire area of the sub-region under it thus providing accurate approximation process though some nodes are not actually present in the sub-region. P_D, parent of node D follows case II of DUMMYREG to regenerate virtual attribute readings at D. Similarly, P_B, parent of node B follows case III of DUMMYREG to regenerate virtual attribute readings at B.

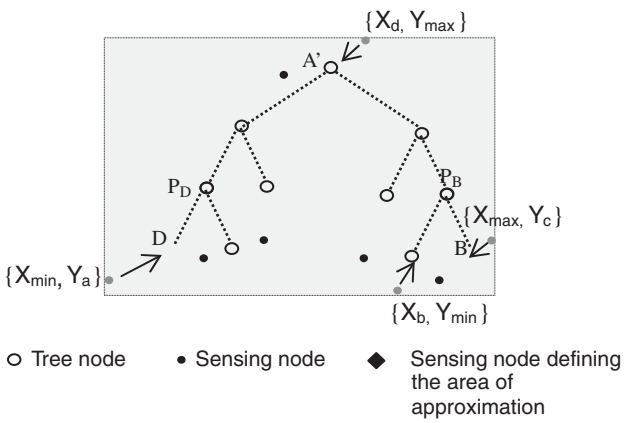

Figure 4. Illustration of how a node of an incomplete QT calculates the boundary of the region for data regeneration.

DOI: 10.1002/dac

Similar to the CR calculated in Section 3.1 for complete binary QTs, for a non-complete binary QT, the total number of tree nodes will be less than $(2^{p+1}-1)$ but will otherwise follow the same expression for CR.

5. UPDATING OF COEFFICIENTS

The work described in Sections 3 and 4 leveraged spatial correlation of data for compression. In this section, we propose a scheme for multiple periods of data aggregation, where the tree nodes do not naively perform regression every time the attribute values change. Instead, they update the previously computed coefficients and send them up the tree, assuming a smooth variation from their previous values. Considering the hardware limitations of low-power sensors, usually, saving battery life of sensor is more important than achieving accuracy of readings, and to implement this, our scheme assumes temporal correlation and allows sensors to perform simpler computations themselves with minimum use of any external software. We describe the process in detail as follows.

Matrix, A_m (in Equation (10) Section 3), is computed only once and for all, for the first time the NT nodes report their readings to their nearest tree node and the leaf nodes send their coefficients to their respective parents. At the start of the data aggregation process when there are no stored coefficients, $\beta_0, \beta_1, \ldots \beta_8$ are calculated from Equation 12.

The complexity of this matrix multiplication process is $< O(m^{2.376})$ (the currently known lowest order of multiplying two matrices each of size, is $O(m^{2.376})$ by Coppersmith and Winograd, [22]) where size of A_m is $m \times m$ and that of F is $m \times 1$. In the subsequent intervals, β_0, $\beta_1, \ldots \beta_8$ are obtained by a simple iterative scheme as follows.

Let g and $g+1$ be the consecutive periods, when the NT nodes report data to the tree nodes. We get an approximate value $\beta_{ap(g+1)}$ of β in the current interval $g+1$, from the β_g (calculated from Equation (12) if $g=0$, else calculated from Equation (13) by updating β_g of the previous interval) in the last interval, g:

$$\beta_{ap(g+1)} = \beta_g + D^{-1} \cdot \Delta \tag{15}$$

where g is the number of observations. D is the diagonal matrix of A_m and of same size and $\Delta = F_{g+1} - F_g$ where Δ is the difference of the two F matrices at time instants $g+1$ and g respectively.

The F matrix is recomputed every time the z (attribute) values change. Accordingly, β needs to be updated according to Equation (15). All entries of the diagonal matrix, D are zeros, except the ones along the diagonal. Thus, the number of multiplications computed in Equation (15) is much less compared to the ones that would have been needed if we use Equation (12) every time to update $\beta_0, \beta_1, \ldots \beta_8$, making the update process relatively faster. In the pseudo-code UPDATE_COEFF (shown in Figure 5), in step 2, TREG is called. When executing TREG, in step 1(b) of Figure 1, instead of performing regression to calculate the new β at $(g+1)$th period, $\beta_{ap(g+1)}$ is stored in $\beta_0, \beta_1, \ldots \beta_8$ by updating β_g, making the time complexity of update $\ll O(m^{2.376})$. The frequency with which F needs to be updated depends on how unstable the network is. In other words, the higher is the change in attribute values with time, greater is the number of times that the coefficients need to be updated.

DOI: 10.1002/dac

PSEUDOCODE *UPDATE_COEFF*

$(z_{1(g+1)}, z_{2(g+1)}, \ldots, z_{n(g+1)})$

Input:
The coefficients of *g*th instant of time.

begin
1. **for** each of the nodes, *k* of the tree of depth, *p*

a. each element of *F(g+1)* is computed as

$$F_{i(g+1)} = \sum_{l=1}^{n_s} z_{(g+1)l} \cdot \phi_i(x_l, y_l)$$

b. β- values of the (g+1) th instant is computed as

$$\beta_{ap(g+1)} = \beta_g + D^{-1} \cdot (F_{g+1} - F_g)$$

endfor

IV. Call $TREG(p, n_s)$

end

Figure 5. Algorithm for updating previous period's coefficients, *UPDATE_COEFF*.

6. CONSTRUCTION OF TREE

6.1. Decision of node at what location should become the root

This section discusses in details, our scheme proposed in Reference [7] to determine the location of the root so that minimum communication overhead is incurred between the sink (which has the provision of being mobile in our scheme) and root. SEAD [23] defines a data dissemination protocol where the sink can be mobile thus keeping the source of the aggregation tree separate from the sink. The SEAD scheme proposes to involve minimum communication energy between sink and the source. Our scheme also keeps root of QT separate from the sink. Which node should become the root (to begin the tree formation algorithm, FORM_QT) is decided by running our proposed DECIDE_ROOT algorithm in a distributed manner. In Figure 6, d is defined as the length of a side of the region approximated by a single QT. The region is divided into a number of square sub-regions. Let l is the length of a side of the square sub-region which houses nodes of a QT. d/l gives the number of such squares along the length (or breadth). The roots of the neighbouring sub-regions need to be situated as close to each other as possible so that minimum number of hops are required among them to send attribute information involving more than one sub-region to sink. The optimal location of each root node (marked by X in Figure 6) is one corner of the square so that neighbouring roots are all bunched together. Selection of the correct corner is crucial and it is a corner on the right edge for odd numbered regions. Similar observations along the breadth help us to identify the relation between a specific sub-region and its preferred corner, which we define as *root intersection*. We allow a variation of a threshold δ from this location.

The reasons for the corner selection are as follows: we assume a general case in which the sink does not have prior information about the root location. Thus, it merely needs to route its query

DOI: 10.1002/dac

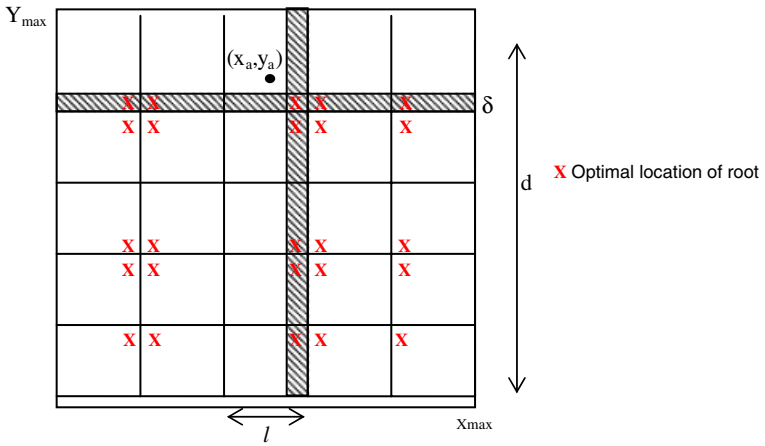

Figure 6. Optimal location of root nodes over the network.

PSEUDOCODE DECIDE_*ROOT* $(X_{max}, Y_{max}, X_a, Y_a)$

Input: the maximum x-y coordinates of the network,
coordinates of the node

begin
if *d*//is even
root_y=call **ROOT**(Y_{max}, Y_a)
root_x=call **ROOT**(X_{max}, X_a)
else
if $Y_a < l$ then
if $|l - Y_a| < \delta$ then root_y=true
else root_y=call **ROOT**(Y_{max}, Y_a)
if $(X_a > X_{max} - l)$ **then**
if $|X_{max} - l - X_a| < \delta$ then root_x=true
else root_x=call **ROOT**(X_{max}, X_a)
end

Figure 7. Algorithm DECIDE_ROOT for deciding root placement.

to the nearest *root intersection* with some certainty of finding a root there. Therefore, for a node at (X_a, Y_a), DECIDE_ROOT (given in Figure 7) and the function ROOT (given in Figure 8) invoked by it, first enables it to determine if it lies within the permissible distance from the y axis of the optimal root location. This is repeated for the *x*-axis, and any node which satisfies both these conditions is eligible to be the root and all such nodes broadcast their eligibility. There should, however, be a single attribute specific root for a sub-region and the final selection is made based on the node ID. We assume that the nodes are aware of (X_{max}, Y_{max}) in this

DOI: 10.1002/dac

PSEUDOCODE ROOT(Max, value)

Input: the maximum value of the coordinate in the network, corresponding coordinate of the node
Output: Boolean value true or false depending on whether the node will be the root or not respectively..
begin
ans=false
n=1
find minimum n such that
$value > (Max - (n-1) \times l)$
if n is even the n
  if $|Max - n \times l - value| < \delta$ then
    ans=true
end

Figure 8. Algorithm ROOT to decide whether the region is odd or even numbered.

Table I. Definition of variables used.

Symbol	Definition
n_s	Total number of sensing nodes reporting to each tree node
p''	Threshold for tree node selection
p	Depth of QT
A_N	Area of the network
A_s	Area of a sub-region containing a single QT
R	Radio range
D	Total number of nodes in the network

algorithm, i.e. the co-ordinates defining the network, l, δ. There are small placement differences when d/l is even or odd and our algorithm, ROOT takes care of these cases as well.

6.2. Construction of query tree

The trees are designed to have a pre-assigned depth, p and hence involve a maximum of p-hops for complete traversal. All the nodes of a tree store the same attribute type. The tree is assumed to be well balanced (complete in the best case) and this results in lower loss of data increasing the accuracy of data aggregation [24]. To facilitate the description of our scheme, we have defined the parameters that we have used, in Table I.

6.2.1. Deriving an upper bound on tree depth, p. We now derive an upper bound on the depth p given the area of the network, A_N and the total number of nodes in the network, D. Density (ρ) of the network is equal to D/A_N. A_s is the area of a sub-region which contains a single compression tree T_c. Therefore, the average number of nodes, S, in the sub-region is given by $\rho \times A_s$. In a complete binary tree, the total number of nodes t can be given by

$$t = 2^{(p+1)} - 1 \tag{16}$$

DOI: 10.1002/dac

Again, assuming that n_s is the average number of sensing nodes reporting to each tree node. Lower bound on

$$S = n_s \times t + t \text{ or } t = s/(n_s + 1) \tag{17}$$

Substituting the value of n is Equation (1); we get an optimal value for the depth of QT

$$p = \mathrm{In}\left(\frac{s}{n_s + 1} + 1\right) - 1 \tag{18}$$

As an example, $D = 1930$, $A_N = 800 \times 800$. Therefore, $\rho = 1930/800^2 = 0.0025$, $A_s = 400 \times 400$. Upper bound on the number of nodes in the region $S = 0.002469 \times 400 \times 400 = 408$.

Assuming depth $p = 4$, number of nodes in $T_c = t = 2^{(4+1)} - 1 = 31$, $n_s = 12$, leads to total number of sensing nodes $= 31 \times 12 = 372$. Therefore, the number of nodes actually in the region S, for TREG to give accurate results $= 31 + 372 = 403 < 408$. Thus the parameters defined above are valid.

Again, n_s is independent of the depth of QT, i.e. the total number of nodes sampled by each tree node is independent of the depth. However, the total number of nodes sampled in the network depends on the depth of QT and is given by

$$n_s^* t$$

As depth of tree of QT is increased, number of approximations increase proportionately thereby deviating more from the true value. To control this, the number of children of QT could be increased further. However, this introduces increase in the computational time at each level due to increased number of children. By simulation, binary tree (each tree node having two children) is shown to exhibit a trade-off between accuracy and computational delay and therefore chosen as the preferred depth throughout the simulations.

6.2.2. Proposed tree formation algorithm. By ensuring that QTs are spread throughout the entire network, attribute readings sent by the sensing nodes (or non-tree nodes, NT) to the corresponding QT incur a smaller hop count. Our algorithm FORM_QT (shown in Figure 9) that creates binary trees of a given depth builds on TREECAST [9], a stateless routing scheme. Each root runs the modified algorithm to create attribute-based trees. Every time a node has to select two of its children, it selects the two nodes farthest apart. This ensures that the tree is widely spread covering as much sensing region as possible, so that the maximum accuracy of the approximation process could be achieved. It also ensures that redundancy in reported attribute values is reduced. As greater area is spanned by the QT, more information about the sensed network is incorporated in the aggregation process, enhancing its accuracy (increased amount of data gives rise to higher correlation and therefore higher accuracy [25]). Thus, our approximation algorithm performs better with increasing number of children at each level and hence incomplete binary trees can be considered as the worst-case scenario to show performance improvement. The tree formation process of TREG is achieved through three types of messages: BEACON, PROBE and JOIN as discussed in the description of the algorithm below. All the message packets contain the sender's and receiver's ID except BEACON message (a broadcast message) which contains just the sender's ID. Figure 10 describes the exchange of different signals to construct the QT. A formal pseudo-code of the algorithm is also included in Figure 9. Three types of control messages are exchanged for tree construction.

DOI: 10.1002/dac

PSEUDOCODE *FORM_QT* (***p,p"***)

Input: the depth and the b-value
Output: a binary tree T_c rooted at r of depth atmost *p* and a unique ID assigned to each node of T_c.
begin
for each level j from *0* to *p-1*
for each node *i* from *1* to 2^j
m_i is a node at level j+*1*
n_i is a node at level j
n_i sends *BEACON* packet containing n_i's ID a_{n_i} to m_i where distance between n_i and m_i<r
m_i chooses n_i as its parent with probability *p'>p"*
m_i sends *PROBE* packet to n_i
n_i waits *NWAIT* time (which is a sufficiently long fixwd time period) to receive *PROBE* pkt from each m_i who selected n_i as parent
end

Figure 9. Tree formation algorithm, FORM_QT.

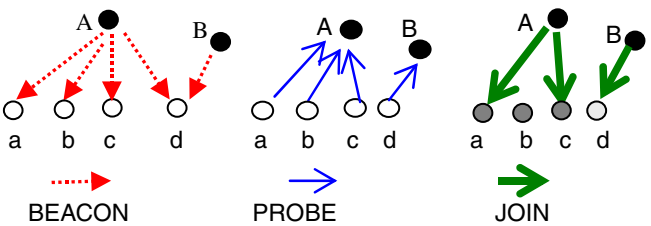

Figure 10. Selection of children by a parent node.

(a) BEACON Message: Each node at level *j* broadcasts a BEACON packet to all its one-hop neighbours. Therefore, a node at level $(j+1)$ can receive multiple BEACON packets from level *j* nodes. It chooses one of them randomly with a probability $p' > p''$ as its parent node and sends a PROBE packet to it. From Figure 10, it is seen that node *d* receives BEACON from both *A* and *B*. The value of p'' (which is an input to the algorithm) is optimized a follows:

Assuming that a parent node *i* broadcasts its BEACON message to a sector of angle 120° and the communication radius to be *r*, the area of this sector $\frac{120}{360} \times \pi \times r^2 \times \rho$.

Therefore, the number of nodes in this region,

$$n_r = \frac{120}{360} \times \pi \times r^2 \times \rho \tag{19}$$

DOI: 10.1002/dac

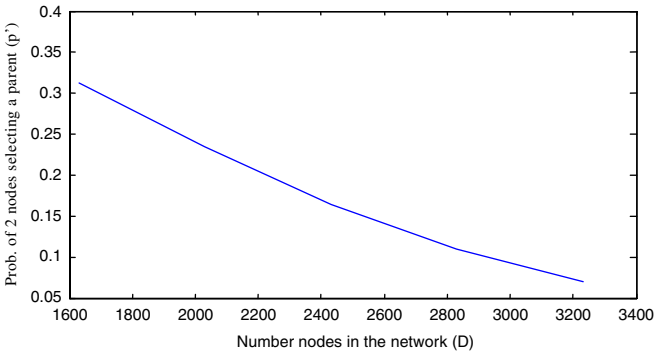

Figure 11. Variation of p'' with no. of nodes D.

Probability p'' that exactly two nodes in this region selects i as their parent follows a binomial distribution.

$$p'' = C_2^{n_r} \times (0.5)^2 \times (1 - 0.5)^{n_r - 2} \tag{20}$$

Figure 11 shows the variation of p'' with number of nodes in the network, D. If a node receives multiple beacons from prospective parent nodes, their IDs are saved for new parent selection for faster recovery in case of node failure. We however leave this as future work.

(b) PROBE Message: The chosen parent waits to receive PROBE packets from all its children before deciding which two are to be selected as children. Once, the parent has heard from all its one-hop $(j+1)$ level nodes, it selects the two nodes farthest apart as its children. The nodes which are not selected by any parent do not try for membership in QT anymore and behave as normal sensing nodes. This ensures proper reduction in the size of the tree (again, frequency of sensing is much greater than frequency of storing). From Figure 10, node d selects node B as its parent and sends PROBE packet to it. Again, node A sends BEACON to nodes a, b, c and d but receives PROBE from nodes a, b and c only.

(c) JOIN Message: After selecting children, the parent sends JOIN message to them, intimating their inclusion in the tree. A however, selects nodes a and c as it children (as is evident from the JOIN message it sends to each of them in Figure 10) since a and c are further apart than either of a and b or b and c.

7. SIMULATION RESULTS

We have simulated our tree construction algorithm, FORM_QT in S*imJava* [26], a discrete event simulation package. We assume a collision free MAC protocol and list the values of the simulation parameters in Table II. As FORM_QT is a modification of the *TREECAST* algorithm [9], its performance is observed to be the same as given in Reference [9]. We thus focus on performance evaluation of our proposed TREG algorithm. A network of 450 nodes is simulated with 12 sensors (n_s) reporting to each tree node and a QT of depth 4. This number of NT nodes is observed to provide a highly accurate representation of the environment. *Mathematica* [27] tool is used for calculating the coefficients at each node during the execution

DOI: 10.1002/dac

Table II. Values of simulation parameters used.

Symbol	Value
A	800×800
R	40 mm
D	1930
$\rho = A/D$	0.0025
As	400×400
p''	0.33
p	4
n_s	12
w	20

of this algorithm. While evaluating the results we consider the following metrics: (1) contour matching between real and approximated data, (2) the accuracy of the approximation of the sensed parameter over the entire region both in absolute value and percentage error, (3) compression ratio, and (4) data packet size at the root node with and without compression. Our simulation study consists of two different data sets: a synthetic model generated in the laboratory based on spatial correlation of attributes and real world data model taken from the data set from rooftop of ATG, University of Washington [28] (relative humidity, temperature, etc.). We first generate a temperature gradation contour over a 2-D region from the real world data [28] for testing the accuracy of our algorithm as shown in Figure 6(i). The temperature attribute shows a change of 1 unit for a traversal of every 45 units. We then place about 400 sensor nodes in an area of 400×400 square units. Over this area, a temperature range of 30–34°F can be realistically assumed for a highly unstable region, like a volcanic eruption site where sensors may be deployed. This assumption of temperature range is validated by the small temperature variation shown in National Observational Data [29] for an undisturbed region. We note that smaller variation in the sensed parameter will only improve the accuracy of our scheme. The first set of results show the effect of the depth of QT on one period of aggregation when we assume that the query has already disseminated to all the leaf nodes of QT.

(1) *Contour plot*: In Figure 12(a) our synthetic model depicts the correlation between sensed attribute values and co-ordinates of NT nodes. Figure 12(b) depicts the same but with approximated values obtained after running TREG at each node of QT. In this plot the recorded temperature readings are in Fahrenheit scale [28]. A comparison of the plots shows that our scheme does not limit the accuracy to certain regions of the contour. The gradation and scales of approximated temperature contour matches the actual temperature distribution in the region almost exactly.

(2) *Percentage error*: Error is calculated as the absolute deviation from the true value and percentage error

$$E = \left(\frac{|z - \bar{z}|}{z} \times 100 \right) \leqslant \varepsilon_{\mathrm{Th}}$$

where $\varepsilon_{\mathrm{Th}} = 6\%$ is the error threshold.

For different depths of the query tree, we obtain similar error levels for both synthetic and real world data. Figure 13 shows the variation in percentage error with the depth of a complete QT. We observe a steady fall in mean error and percentage error for synthetic data as the depth

DOI: 10.1002/dac

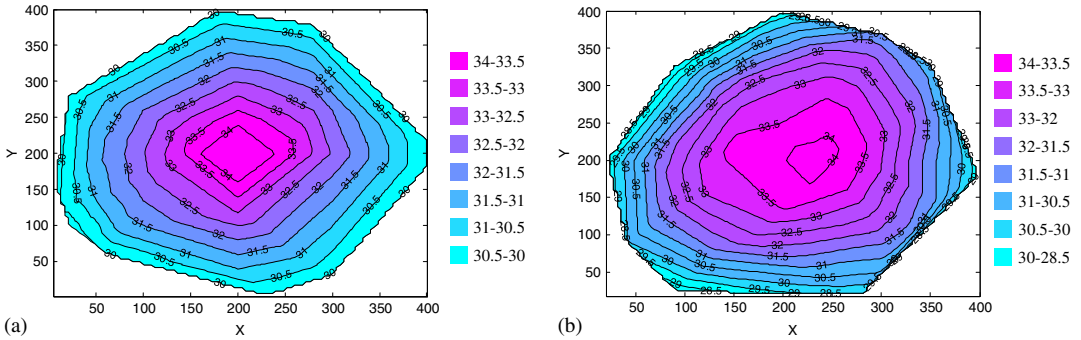

Figure 12. (a) Contour plot of our synthetic model in a 400×400 network; and (b) contour plot of the attribute in the same region obtained after simulation.

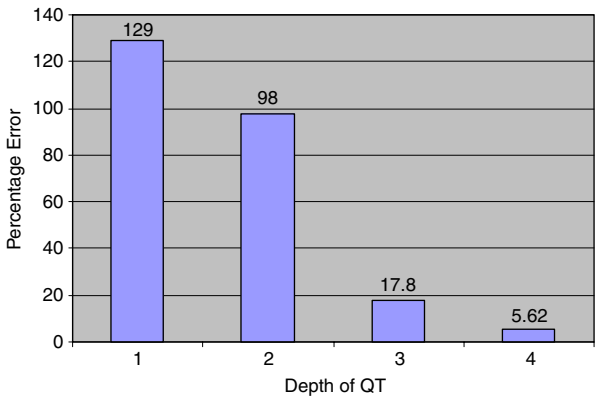

Figure 13. Variation of percentage error with depth.

of the query tree increases. This is expected, as with increase in depth, the tree nodes are able to cover larger area thus being able to approximate the sensed parameter over the region better. A tree with depth 1 gives maximum error, as a small tree of only three nodes (most sensing nodes are dispersed in the region, not within range of tree nodes and are hence unable to report to the QT) is made to approximate the entire region of area A_s. Figure 14 shows the gradation of percentage error upto the maximum value of 5.64 for a tree depth 4, which has least error as shown in the previous result. We take 381 readings and find that the first three bars concentrate the maximum number of points where the approximation is evaluated. These bars show that while the tree of depth 4 had a maximum error of 5.64%, a majority of nodes had their individual error limited to a much smaller level, 0–1.68%. This error distribution has a steadily decreasing nature implying that as error range increases, the number of nodes having a magnitude of error that falls in that range decreases.

Figure 15 shows the variation in percentage error as a function of the depth and the type of tree, i.e., complete and non-complete binary trees. The first bar in each set shows the error incurred when TREG is applied on complete binary tree of varying depths. The best-case error

DOI: 10.1002/dac

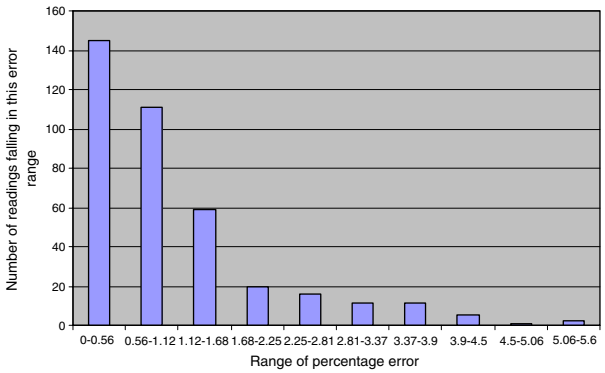

Figure 14. Distribution of error for tree of depth 4.

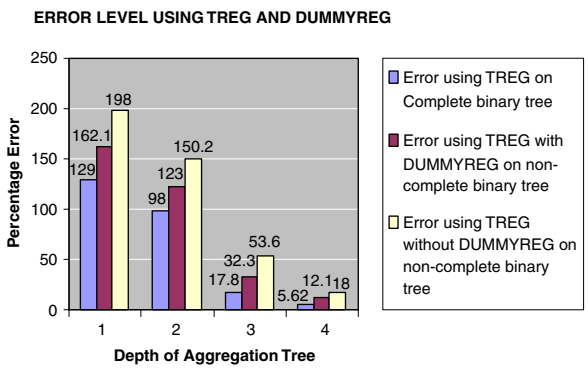

Figure 15. Variation of percentage error with depth and tree type.

is 5.62%. The second bar indicates the error incurred when TREG is applied to non-complete binary trees of varying depth by calling DUMMYREG. In this case the best-case error is 12.1%. The third bar shows the corresponding errors (worst case error is 21%) when TREG is applied to non-complete binary trees without calling DUMMYREG. Thus we see that without the dummying of nodes, the entire region is not approximated accurately, giving larger errors than that the case of full dummying of the sensed region. For a depth of 4 of the non-complete query tree, we obtain different error levels, respectively, for three cases (i) no dummying, (ii) partial dummying and (iii) full dummying. In the case of no dummying, DUMMYREG is not called from TREG at all. In the case of partial dummying, the corrective cases I and II of DUMMYREG are not applied to the topology thus still incurring greater error. Full dummying applies all the corrective measures of DUMMYREG thus giving the best-case error. From Table III, we observe a steady fall in percentage errors different levels of dummying are applied. Without dummying, readings from locations without tree nodes do not participate in the DUMMYREG algorithm. By dummying the otherwise absent nodes of the tree, attribute values are generated at those imaginary nodes and their participation in the compression method provides better approximate readings of the entire sensed region,

DOI: 10.1002/dac

Table III. Different degrees of dummying.

Cases	Error level (%)
No dummying	21
Partial dummying	18
Full dummying	12

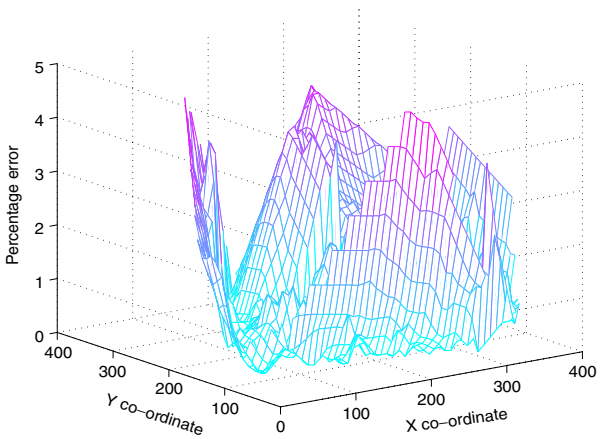

Figure 16. 3-D error plot for depth 4.

instead of only the portions where sensors are actually present. Thus, when attribute values are regenerated at the root spanning the entire region, the non-dummying case produces attribute values at some locations highly uncorrelated form the true values since readings from these locations were not taken into account by the parent nodes due to absence of sensors (children nodes). But, the dummying case gives better results when values are regenerated back at the root as readings from the entire spanned region have been reflected by the approximation process. This reduces the error level considerably, improving our approximation scheme in terms of its accuracy.

(3) *3-D error plot*: Figure 16 shows 3-D percentage error plots for a complete binary QT of depth of 4. The error plots have a concave shape with peaks at the border. This is because the tree structure is concentrated at around the central areas of the sub-region and nodes at the extremities do not report to the query tree. The approximation at the edge of the region is greater and this results in higher error values as is shown in the plot.

(4) *Compression ratio*: Figure 17 shows the variation of compression ratio with depth of the tree and as expected it is found to be almost constant giving a value of 0.02. The descending nature of the curve in Figure 17 suggests that with increase in depth, compression ratio expressed as the output data size as a fraction of the input data size, decreases. This is expected as greater is the depth, better is the degree of compression as further reduction in output data occurs. A high rate of compression reduces the overall message size thus saving the overall communication bandwidth. Also, a constant compression ratio is preferred as it makes it easier to model the hierarchical structure of the QT.

DOI: 10.1002/dac

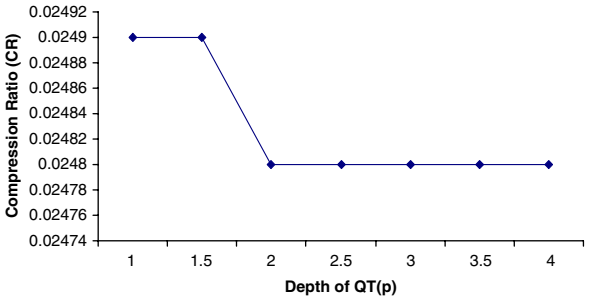

Figure 17. Dependence of compression ratio on depth of QT.

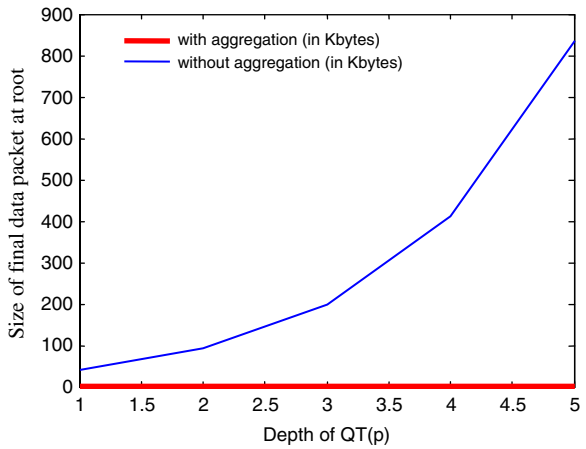

Figure 18. Dependence of size of data packet (at root node) on depth of QT.

(5) *Output data packet size at the root*: Figure 18 shows the size of the data traffic at the root with data aggregation by TREG and for normal many-to-one communication where the root receives all the data without compression. For different depths of QT, the size of the data packet transmitted by the root to sink is constant when data compression is performed by all the nodes of QT and is of fixed size $(s_x + s_y + s_c)$ bytes. The data packet size transmitted from one tree node to another is fixed and is independent of the network size, thus keeping the overall energy for transmission of data within reasonable bounds. This validates our assumption that every tree node transmits to its parent, a data packet which contains only the coefficients and the range of x and y-co-ordinates. This data packet size is independent of the number of nodes in the tree and therefore the sub-region. In traditional many-to-one communication, all leaves have to send the attribute readings to the root. Therefore, the size of the data packet to be transmitted by the root node increases without any bounds with increase in the network size. Performing data compression with TREG, data traffic is reduced by 95% as compared to Reference [30] which achieves a maximum data traffic reduction of 85%.

DOI: 10.1002/dac

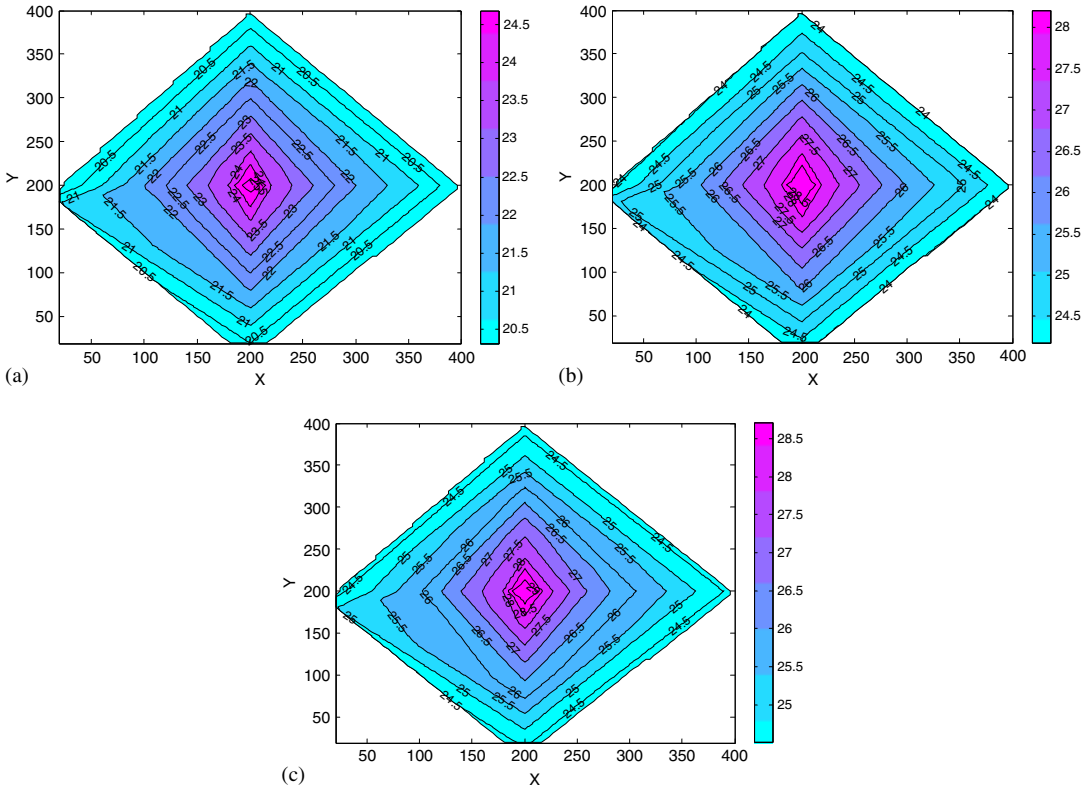

Figure 19. (a) Temperature contour at instant g; (b) actual temperature contour at instant $g+12$; and (c) approximate temperature contour at instant $g+12$, by regenerated.

The next set of results collected for multiple periods of data aggregation, shows the performance of the approximation process when coefficients obtained from the preceding rounds are updated using UPDATE_COEFF.

To validate UPDATE_COEFF in our simulation, real world data (from the 24-h temperature record on 24 June as Figure 71-3 of Reference [31]) is placed on a synthetic model of grid 400×400 structure (see Figure 19(a)). This set of simulations are done in Matlab [32] with parameters used given in Table II, to generate the contour model at a later instant, given a previous instant.

(6) *Contour plot using UPDATE_COEFF*: At instant $g=0$, the temperature distribution is obtained from Figure 71-3 of Reference [31] as shown in Figure 19(a). Figure 19(b) shows the contour plot of the same region at time instant $g+12$ (after 22 h with a sampling period of 2) sensing the actual temperature in the range 24–28°C, when a more or less small yet steady increase in the overall temperature has occurred. Figure 19(c) shows the contour plot of the regenerated temperature values for QT of depth of 4 when the final set of attribute values are regenerated by using β_{g+12} obtained from Equation (13). The values of $\beta_{ap(g+12)}$ are obtained by recursively updating β_g in Equation (13) 12 times. β_0 is obtained by performing regression using Equation (12). In this plot, we measure the temperature in Centigrade scale. A comparison of the plots shows that updating the coefficients (returning $\beta_{ap(g+1)}$) does similar kind of

DOI: 10.1002/dac

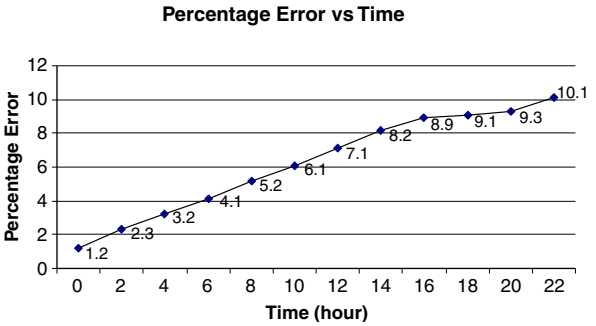

Figure 20. 3-D variation of percentage error with interval of time.

approximation as done by performing Gaussian elimination in Equation (12) (giving β_{g+1}) 11 times, every time the attribute values change. Yet the updating process requires $\ll O(m^{2.376})$ number of computational steps as compared to Gaussian elimination having worst-case complexity of $O(m^3)$.

(7) *Percentage error*: Figure 20 validates the conclusion drawn by analysing Figure 19(b) and 19(c) by calculating the error as the absolute deviation from the true value and is expressed as percentage error. The percentage error is calculated by updating the coefficients every 2 h over a 24 h period. Thus, the coefficients obtained at gth interval is updated 12 times to get the final set of coefficients reflecting the temperature of the region at 23rd hour of the day. In other words, we consider a sliding window (i.e. the number of most recent data values recorded by the sensor) of size 12. The percentage error is observed to be strictly increasing with time since increasing approximation is done with increase in time interval. Until the first five updating steps (till 10th hour of observation), the error is 5.3% ($<\varepsilon_{\mathrm{Th}}$) and at the same time, needs to simply update the coefficients using UPDATE_COEFF at every node thus saving computation time and battery life of the low-power sensors. The coefficients of current interval are similar to that of the previous interval because of the temporal correlation of sensed attribute.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel data aggregation strategy in environments which exhibit naturally occurring phenomenon, by building multiple attribute-based trees based on function approximation. This scheme allows measurement of sensed parameters at locations devoid of sensor nodes. Also by restricting the number of messages and keeping constant message size irrespective of the depth of the aggregation tree, we obtained a percentage error that was within acceptable limits. Our scheme is scalable as the data compression ratio is nearly constant and accuracy increases with increasing density. Proposed tree construction phase enables distributed root selection and obviates the need for global root location information at the sink. Our proposed scheme considers the property of temporal correlation of sensors to regenerate attribute values over the sensed region. Over a long period of time, the computational complexity incurred in getting the new attribute values every time they change is significantly low. TREG is observed to give substantially higher percentage error of the order of 30% in environments where parameters do not follow a smooth correlation (image contour analysis

DOI: 10.1002/dac

with random pixel values). TREG works with the assumption that sensors are giving accurate results. If the entire network is distorted, a bulk of the nodes report wrong readings, thus TREG fails to detect these to be wrong readings. However, if a localized area of readings is faulty and the corresponding tree node receives readings only from faulty NT nodes, its coefficients sent to its parent will be significantly different from its sibling. The parent can infer from the regenerated data whether this is part of a new event or faulty readings by observing that the deviation is greater than a factor, T_h, i.e. $|i_{g+1}-i_g| \geqslant T_h$ where i_{g+1} is a data value reported by a NT node i at the instant $g+1$ and i_g is the data reported by the same node in the previous instant. Again, there will be tree nodes which receive readings from both faulty and as well normal nodes. In that case, the corresponding tree node will compare this reading with the other neighbouring tree nodes and observes that this reading is significantly different from any of its neighbours. Work is going on currently on this issue of detection of single point failures and localized faults and also on how to detect event boundaries based on further analysis of the final polynomial obtained at the root. Future work will include computing the energy required for communication and latency involved per round of data aggregation. The maintenance of the tree structure in the event of node failure is a likely step to improve robustness. If the network gets partitioned in two parts, each part can calculate its own set of coefficients at each of the roots of the individual QTs. When connection restores, the individual coefficients and the x–y range spanned by each of the sub-networks can be used to regenerate new data points and new set of coefficients which gives a globally optimal solution. However, this claim needs to be validated and will be addressed in our future work. We intend to apply our proposed scheme to an actual sensor test-bed for validation and observe the actual savings in energy and communication time. Future work would also involve the application of our proposed algorithm in a multi-attribute scenario and study of how frequently the coefficients need to be updated based on the dynamicity of the network.

ACKNOWLEDGEMENTS

This work has been supported in part by the Ohio Board of Regents' Doctoral enhancement Funds. The authors take pleasure in thanking Dr Donald French, Assistant Professor, Department of Mathematical Sciences, University of Cincinnati, for many helpful discussions.

REFERENCES

1. Agrawal D, Zeng Q-A. *Introduction to Wireless and Mobile Systems*. Brooks/Cole Publishing, 2003; 436. ISBN 0534-40851-6.
2. Intanagoniwiwat C, Estrin D, Govindan R, Heidemann J. Impact of network density on data aggregation in wireless sensor networks. *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, July 2002; 575–578.
3. Heinzelman W, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks. *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, Haui, HI, January 2000.
4. Ferriere HD, Estrin D. Efficient and practical query scoping in sensor networks. *Technical Report 39*, CENS, UCLA, Los Angeles, CA, U.S.A., April 2004.
5. Kuh E. *An Essay on Aggregation Theory and Practice*, *No. 43*. Working papers from Massachusetts Institute of Technology (MIT), Department of Economics.
6. Sharifzadeh M, Shahabi C. Supporting Spatial Aggregationin Sensor Network Databases. *GIS'04*, Washington, DC, 12–13 November 2004.
7. Banerjee T, Chowdhury KR, Agrawal DP. Tree based data aggregation in sensor networks using polynomial regression. *Proceedings of the 8th Annual Conference on Information Fusion*, Philadelphia, 25–29 July 2005.

DOI: 10.1002/dac

8. Banerjee T, Chowdhury KR, Agrawal DP. Distributed data aggregation in sensor networks by regression based compression. *Proceedings of the 1st International Workshop on Resource Provisioning and Management in Sensor Networks (RPMSN05) in Conjunction with MASS*, Washington, DC, 7–10 November 2005.
9. PalChauduri S, Du S, Saha A. Johnson D. Tree cast: a stateless addressing and routing architecture for sensor networks. *4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks, WMAN 04*, Santa Fe, NM, 2004.
10. Kotidis Y. Snapshot queries: towards data-centric sensor networks. *Proceedings of ICDE*, Tokyo, Japan, 2005.
11. Cristescu R, Vetterli M. On the optimal density for real-time data gathering of spatio-temporal processes in sensor networks. *Proceedings of IPSN*, Los Angeles, California, 2005.
12. Vuran MC, Akan OB, Akyildiz IF. Spatio-temporal correlation: theory and applications for wireless sensor networks. *Computer Networks Journal (Elsevier Science)* 2004; **45**(3):245–259.
13. Stronger D, Stone P. Polynomial regression with automated degree: a function approximator for autonomous agents. *Technical Report UT-AI-TR-06-329*, AI Laboratory, The University of Texas at Austin, Department of Computer Sciences, 2006.
14. Guestrin C, Bodix P, Thibaux R, Paskin M, Madden S. Distributed regression: an efficient framework for modeling sensor network data. *Proceedings of IPSN '04*, Berkeley, California, 2004.
15. Meng X, Nandagopal T, Li E, Lu S. Contour maps: monitoring and diagnosis in sensor networks. *Computer Networks Journal* 2006; **50**(15):2820–2838.
16. Solis I, Obraczka K. In-network aggregation trade-offs for data collection in wireless sensor networks. *INRG Technical Report 102*, 2003.
17. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. *IEEE Communication Magnatic* 2002; **40**(8):102–114.
18. Langendoen K, Reijers N. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks: The International Journal of Computer and Telecommunications Networking (Special issue: Wireless Sensor Networks)* 2003; **43**(4):499–518.
19. Borgelt C. *Intelligent Data Analysis*. School of Computer Science Otto-von-Guericke-University of Magdeburg, Universiťatsplatz 2, D-39106 Magdeburg, Germany.
20. Finn JD. *A General Model for Multivariate Analysis*. Holt, Rinehart & Winston: NY, 1974. ISBN: 0-03-083239-X.
21. Fausett L. *Numerical Methods: Algorithms and Applications*, 0130314005 (Hardback). Pearson Education: NJ, October 2002.
22. Coppersmith D, Winograd S. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation* 1990; **9**:251–280.
23. Kim HS, Abdelzaher TF, Kwon WH. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. *Proceedings of Sensys*, Los Angeles, California, 2003.
24. Yuan W, Krishnamurthy SV, Tripathi SK. Synchronization of multiple levels of data fusion in wireless sensor networks. *IEEE GLOBECOM*, San Francisco, California, 2003.
25. Servetto SD. Sensing lena-massively distributed compression of sensor images. *Proceedings of the IEEE ICIP*, Barcelona, Spain, 2003.
26. Simjava [Online] www.dcs.ed.ac.uk/home/simjava/doc/index-all.html
27. Mathematica [Online] www.wolfram.com
28. ATG rooftop data [Online] www.atmos.washington.edu/cgi-bin/uw.cgi?20050223
29. ATG [Online] www.atmos.washington.edu/data/difax/maxtemp
30. Wolf T, Choi SY. Aggregated hierarchical multicast for active networks. *MILCOM* '01, Mclean, VA, 2001.
31. http://www.physicalgeography.net/fundamentals/7l.html
32. Matlab [Online] http://www.mathworks.com

AUTHORS' BIOGRAPHIES

Torsha Banerjee received the BTech degree in computer engineering from University of Kalyani, West Bengal, in 2003. She is currently a PhD student in computer science and engineering at the Center for Distributed and Mobile Computing Laboratory, University of Cincinnati. Her research interests include wireless ad hoc networks, wireless sensor networks with focus on data aggregation and fault tolerance, and spectrum management in cognitive radio. She is a student member of IEEE.

DOI: 10.1002/dac

Kaushik Chowdhury received his BE degree in electrical and electronics engineering from VJTI, Mumbai University, India, in 2003. He received his MS degree in computer science from the University of Cincinnati, OH, in 2006, graduating with the best thesis award. He is currently a Research Assistant in the Broadband and Wireless Networking Laboratory and pursuing his PhD degree at the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA. His current research interests include multichannel medium access protocols, dynamic spectrum management, and resource allocation in wireless sensor networks.

Dharma P. Agrawal is the Ohio Board of Regents Distinguished Professor of Computer Science and Engineering and the founding director for the Center for Distributed and Mobile Computing in the Department of ECECS, University of Cincinnati, OH. His current research interest includes energy efficient routing and information retrieval in mesh, ad hoc and sensor networks, QoS in integrated wireless networks, use of smart multi-beam directional antennas for enhanced QoS, and secured communication in mesh, ad hoc and sensor networks. His co-authored textbook on *Introduction to Wireless and Mobile Systems*, published by Thomson has been adopted throughout the world and revolutionized the way the course is taught and the second edition has been published recently. His latest co-authored book Ad hoc & *Sensor Networks—Theory and Applications* has been published in March 2006 by the World Scientific Publishing. He is an editor for the *Journal of Parallel and Distributed Systems, International Journal on Distributed Sensor Networks, International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), International Journal of Ad Hoc & Sensor Wireless Networks, and the Journal of Information Assurance and Security (JIAS)*. He has served as an editor of the IEEE *Computer magazine*, the *IEEE Transactions on Computers* and the *International Journal of High Speed Computing*. He has been the Program Chair and General Chair for numerous international conferences and meetings. He has received numerous certificates and awards from the IEEE Computer Society. He was awarded a '*Third Millennium Medal*,' by the IEEE for his outstanding contributions. He has also delivered keynote speech for six international conferences. He is a Fellow of the IEEE, the ACM, the AAAS, and WIF.

DOI: 10.1002/dac