

Supporting Topographic Queries in a Class of Networked Sensor Systems*

Mitali Singh and Viktor K. Prasanna
Department of Computer Science
University of Southern California
Los Angeles, CA-90089.
{mitali, prasanna}@usc.edu

Abstract

Topographic querying is the process of extracting data from a sensor network for understanding the graphic delineation of features of interest in a terrain. Query processing techniques based on localized computation and aggregation cannot be used to resolve topographic queries efficiently as resolution of such queries involves global collaboration among the sensor nodes (e.g., region identification). We explore an alternate approach for resolving such queries, which is based on construction and maintenance of topographic maps. We focus on the class of sensor systems involving dense, uniform deployment of sensor nodes over a two dimensional terrain. Our analysis shows that by maintaining additional state (map) in the system, the query processing costs can be reduced significantly in comparison with the state-of-the-art. Our simulation results show that the map construction overheads are amortized over a small number of queries.

1. Introduction

Topographic querying is the process of extracting data from a sensor network for understanding the graphic delineation of features of interest in a monitored terrain. Application areas where such information is particularly useful include contaminant monitoring, tracking soil moisture levels, water temperature estimation of river beds, etc. [8, 15]. A prototype implementation of a sensor network used for constructing topographic maps of soil moisture and temperature in vineyards is described in [1]. The Sensor Web project at NASA/JPL [16] describes small-scale deployment of sensor networks for understanding spatio-temporal aspects of dynamic phenomenon in habitat monitoring and microsensing applications.

Topographic queries can be used for extracting a wide variety of topographic information from the network. The end user might be interested in periodically obtaining the total area spanned by a set of feature regions in the network or estimating the boundary of a particular region. Other useful topographic queries of a smaller scope include enumeration and/or description of regions with sensor readings in a specific range [10]. Boundary estimation and counting regions of interest can be used in acoustic monitoring and tracking applications [13]. For example, a spatio-temporal visualization of acoustic readings can be used to determine the number and locations of targets in the region [3]. Topographic information about properties such as residual energy levels are useful for resource management, dynamic retasking, preventive maintenance of sensor networks, etc.

In recent years, several research efforts have focused on design of localized protocols for sensor networks that abstract the network as a “stateless” system. We argue that this approach is not suited to all sensor applications. Several applications will benefit both in terms of cost efficiency and computability if some global state is maintained in the system. In this paper, we explore a “stateful” approach for resolving topographic queries. These queries have the following characteristics. As opposed to geographical locations, the queries are routed towards feature regions in the network. Results of such queries cannot be locally computed at individual nodes, but require global collaboration involving distant nodes in the network (e.g., boundary detection of feature regions). Thus, state-of-the-art techniques involving query-initiated explicit data aggregation and exfiltration incur large communication costs in resolving these queries.

Our approach is based on prior construction and maintenance of the *topographic map* of user-specified features in the network. We assume that the features in the network do not change very frequently. We discussed a distributed algorithm for construction of topographic maps in [14]. In this paper, our goal is to demonstrate that once constructed, the map can be used to resolve several topographic queries in an efficient and simple manner. We define and analyze two

* This work is supported by the DARPA Power Aware Computing and Communication Program under contract no. F33615-02-2-4005 and in part by the NSF under grant number IIS-0330445.

types of topographic queries: (a) global-aggregate queries and (b) region-specific queries. Our analysis shows that these queries can be resolved more efficiently by using our approach as opposed to the state-of-the-art techniques.

The rest of the paper is organized as follows. Other related research is discussed in the following section. We define the system model in Section 3, followed by a brief overview of topographic maps. In Section 4, we discuss resolution of several topographic queries using our approach and the state-of-the-art techniques. We present our simulation results in Section 5 and conclude in Section 6.

2. Related Work

A lot of work has been done on query processing in sensor networks [6, 10]. However, most research efforts have focused upon supporting spatial and/or aggregate queries. The basic approach involves construction of a “well-structured” spanning tree in the network which can be used to route the query and the responses efficiently in the network. Localized aggregation/compression techniques are exploited to reduce data communication in the network [10]. Query routes are optimized by using cached route information at nodes and spatial information stored in query packets [7].

Our focus is on resolving topographic queries in the network. As opposed to geographic locations, these queries extract information about feature regions in the network. Resolution of such queries requires global collaboration among the sensor nodes in the network (e.g., region identification and boundary estimation). The queries extract information about feature regions in the network as opposed to geographical locations. In [14], we presented a distributed algorithm for construction of topographic maps in sensor networks. In this paper, we demonstrate how the information stored in topographic maps can be used for efficiently routing and resolving topographic queries. Note that our approach to query-resolution is not “stateless” as in the above discussed state-of-the-art techniques but assumes maintenance of global state in the system.

The work perhaps most closely related to our research is [10], which addresses the feature extraction problem in sensor networks. The proposed technique organizes the network into a forest of region-specific spanning trees. We compare this approach to our approach in Section 4. In [12], the authors describe a state-centric programming model for sensor networks, which will facilitate specification and implementation of our algorithms.

The topographic map construction algorithm presented in [14] exploits node coordinates to overlay a pyramid structure over the network to support hierarchical information storage. A similar infrastructure is used by the DIMENSIONS [4] data handling architecture to support multi-

resolution data access and spatio-temporal pattern mining. However, the focus of the DIMENSIONS architecture is on computation, storage and retrieval of spatial and temporal data aggregates in the network and not on construction and maintenance of topographic maps. Map construction involves identification of feature regions in the network, and extraction and storage of topographic information about the identified regions (such as the boundary, area, and the average feature value). In [6], the authors briefly discuss contour mapping in sensor networks, but do not describe the isobar extraction and merging algorithms in detail.

3. Preliminaries

3.1. System Model

We abstract the sensor network as a distributed system (networked sensor system) consisting of a homogeneous set of uniquely-identifiable sensor nodes that communicate with each other asynchronously over locally shared wireless channels. The key assumptions of our model are discussed below.

1. We assume that n sensor nodes are uniformly distributed over a two dimensional terrain. The sensor nodes are uniquely identifiable (e.g., by their geographical coordinates). Each sensor node is equipped with a low power processor, a memory of size $O(\log n)$ bits, a local clock and a wireless radio with fixed range r .
2. The system is represented by a graph $G(V, E)$, where V is the set of vertices representing the sensor nodes and E is the set of edges. Edge $(u, v) \in E$ exists between two vertices u and v provided they lie within the communication (radio) range of each other. All edges in the graph are symmetric. The sensor nodes that share an edge are called neighbors. The node distribution is sufficiently dense to ensure that the graph is connected. However, we assume that the graph has a small (constant) degree and each sensor node maintains a list of all its neighbor IDs.
3. No global clock exists in the system. Sensor nodes communicate asynchronously over the wireless channel using message passing. Each message is of size $O(\log n)$ bits. Two types of messages are supported in the system: broadcasts and unicasts. A broadcast is a one-to-many message from the sender node to all its neighbors. A unicast is a one-to-one message destined to a specific neighbor.
4. The wireless channel is locally shared - a sensor node can receive a message provided only one of the nodes within its radio range (neighbor) transmits at a time. A medium access mechanism is supported in the system for resolving channel contention. When multiple sensor nodes want to access a shared channel at the same time, the medium access mechanism schedules the communication in conflict-free steps.

All messages are thus reliably transmitted to the neighboring nodes in a small (constant) number of steps.

5. Sensor nodes have two power states: active and switched off. The sensor nodes must be active to perform any computation or communication. A low-power paging mechanism is supported in the system, using which a node can activate other nodes within its radio range.

Communication time: As is customary in the analysis of asynchronous distributed systems [11], for the sake of analysis only, we use a logical clock. We assume that all sensor nodes operate at the same clock frequency and there exists a logical global clock in the network. One time step corresponds to a cycle of the logical clock. We assume that communication of a message between neighboring nodes takes one time step. Paging or state transition overheads are assumed to be negligible. The time complexity of an algorithm represents its overall execution time as measured by the logical clock.

Communication energy: We define one unit of energy as the energy dissipated by a sensor node in transmission or reception of one message over the wireless channel. We assume that communication of a single message between neighboring sensor nodes dissipates one unit of energy at the sender and each receiver node. The energy complexity is defined as the sum of the transmission and reception energy over all the sensor nodes in the system.

3.2. Definitions

A *feature* is a user defined predicate on sensor data. Consider a sensor network deployed for measuring the temperature of the environment. An example of a feature for this network is “is temperature greater than forty?”. A sensor node is said to satisfy a feature, if the result of applying the corresponding predicate on the sensor data is true. The sensor data used for determining whether a given feature is satisfied or not is called the *feature value* of the node, and a sensor node that satisfies the feature is called a *feature node*. All sensor nodes with temperature readings greater than forty are the feature nodes for the above example feature. The temperature

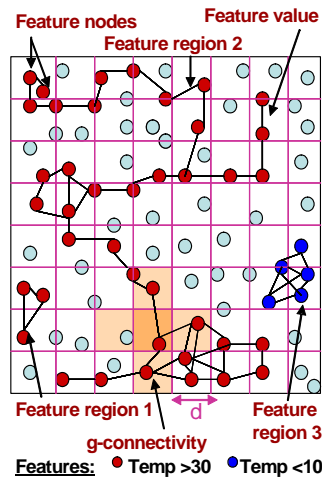


Figure 1. Feature regions

readings stored at the the sensor nodes are the feature values. We assume that the total number of features defined in the network is a constant. Each feature partitions the network into a set of feature nodes and non-feature nodes. A sensor node can locally determine whether it satisfies a given feature or not.

A *feature region* refers to a geographically contiguous area in the network consisting of sensor nodes that satisfy the same feature. Given a feature f , the feature region is defined as follows. Consider a partition of the network into a grid consisting of cells of size $d \times d$ as illustrated in Figure 1. Two sensor nodes are said to be *g-connected* (geographically contiguous) if they belong to the same or neighboring (north, south, west, east) cells. Let $G_f = (V, E_f)$ denote the *feature graph* for f . The vertices $v \in V$ represent the sensor nodes, and an edge $e(u, v) \in E_f$ exists between two vertices u and v iff they are g-connected and they both satisfy feature f . Each connected component in G_f represents a feature region. Here, d defines the resolution factor for region identification. Our assumption about dense deployment ensures that $d \leq r$, implying that nodes in neighboring cells are also wireless neighbors.

readings stored at the the sensor nodes are the feature values. We assume that the total number of features defined in the network is a constant. Each feature partitions the network into a set of feature nodes and non-feature nodes. A sensor node can locally determine whether it satisfies a given feature or not.

3.3. Topographic Map

A *topographic map* represents the graphic delineation of features of interest in a terrain. In [14], we present a distributed algorithm for constructing topographic maps in sensor networks. The algorithm constructs a hierarchical information storage infrastructure in the network for maintaining the topographic information in a distributed manner.

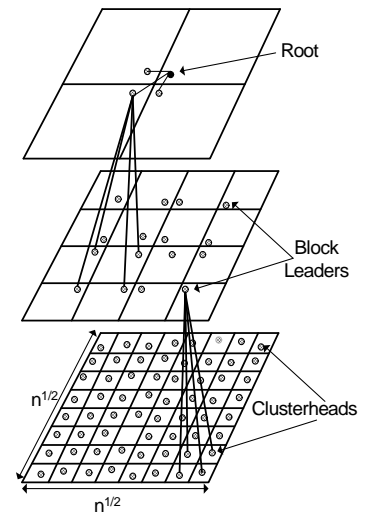


Figure 2. Hierarchical storage

Hierarchical information storage: The network is partitioned into single-hop clusters of size $s \times s$, where $s = \frac{r}{\sqrt{5}}$. A clusterhead is elected for each of the clusters. The above cluster size ensures single hop connectivity between clusterheads in (north, south, west and east) adjacent clusters. Next, the clusterheads organize themselves into blocks of size 2×2 clusters and one of them (closest to the center of the network) is elected as the block leader. This is repeated

recursively for $k = (\log \frac{\sqrt{n}}{s} - 1)$ iterations¹. At the l^{th} iteration, where $l \leq k$, the block leaders at level $l - 1$ organize themselves into blocks of size $2^l \times 2^l$ and elect one of themselves to be the leader of the larger block formed. On the last iteration, one of the block leaders is elected to be the **root** of the network. The network is thus organized into a pyramid structure as illustrated in Figure 2.

Information storage: The map construction algorithm discussed in [14] identifies and labels all feature regions in the network and assigns a parent for each region. The *parent* of the region is defined as the leader of the smallest block that fully contains the region. A feature region is labeled by the id of the minimum id feature node in the region. The algorithm also computes various aggregates over feature values stored at the sensor nodes (such as min, max and average). This information is stored in the network in the following manner.

The clusterheads maintain detailed information about all the feature nodes in the cluster such as the id, location and feature(s) satisfied by the feature node; the label and parent of the feature region(s) to which the node belongs; and whether the node is a boundary node or minimum id node for any feature region. If the node is the minimum id node for its region, the clusterhead also maintains the sum, min, max, and average of the feature values in the corresponding region. Since the total number of features is a constant and there are at most $\frac{r^2}{5}$ feature nodes in each cluster, $O(r^2)$ words of memory is required at the clusterheads, where one word consists of $\log n$ bits.

The block leaders maintain less-detailed, but more global information about the features in the network. For each feature, the following data is maintained: the count of the feature regions and feature nodes that lie within the block, the maximum, minimum, sum and average of the feature values stored at the feature nodes that lie within the block. Block leaders require an additional storage of $O(1)$ words.

3.4. State-of-the-art

We consider the following state-of-the-art techniques as the baseline for comparing our results.

Localized approach [7]: No global state is maintained about the feature regions in the network. The sensor nodes can locally compute whether they satisfy a feature based on the feature values stored by them.

Forest of trees [10]: Feature regions are identified and each region is labeled by the minimum id feature node in the region. A spanning tree is constructed within each region (consisting only of feature nodes that belong to the region), with the minimum id node as the root.

The root node maintains aggregate information about the feature region it represents such as the total number of feature nodes in the region and the minimum, maximum, and average of the feature values over all sensor nodes in the region.

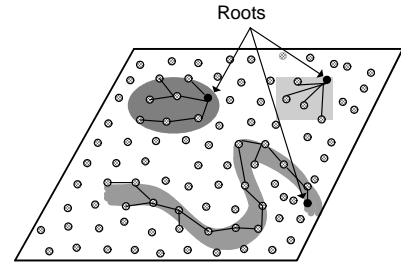


Figure 3. Forest of trees

4. Topographic Queries

In this section, we analyze the time and energy costs for executing topographic queries in the network using our approach and the state-of-the-art techniques. We assume that a query can originate at any sensor node in the network, which is called the *query node*. We discuss resolution of the following two types of topographic queries.

Global aggregate queries: These queries are executed for obtaining system-wide, global aggregates about features of interest in the network. The input to these queries is f , a user defined feature. The query is executed over all sensor nodes that satisfy f . The output is either the result of a computation performed on the feature values stored at the sensor nodes or a label used for identifying a feature region in the network. Some examples of such queries include the following. How many sensor nodes in the network satisfy feature f ? Count the number of feature regions in the network consisting of sensor nodes that satisfy feature f . Compute the weighted average of the feature values stored at the sensor nodes that satisfy feature f (e.g., higher weight given to feature values stored at sensor nodes in the larger regions).

Region specific queries: Region-specific queries extract information about feature nodes belonging to particular feature region in the network. The input to these queries is l , the label used to identify a specific feature region in the network, and the output is the result of a computation performed over all feature nodes belonging to the specified region. These queries cannot be resolved using the centralized approach as this approach does not maintain region labels in the network. Region specific queries are often executed in conjunction with the global aggregate queries. For example, a global aggregate query is executed in the network to obtain the label of the largest feature region in the network. This is followed by a set of region specific queries used for extracting detailed topographic information about the largest region in the network. Some examples of such queries include the following. How many feature nodes in region l have feature values above a given threshold. What

¹ All logs are to the base 2 unless specified otherwise.

percentage of feature nodes in region l have feature value below the average feature value in the region? Compute the median of all feature values in region l .

1. Global aggregate query: Count the number of feature regions in the network consisting of sensor nodes that satisfy feature f .

Application scenario: Consider the target counting problem discussed in [3]. A sensor network is deployed for monitoring the spatio-temporal distribution of the acoustic signals in the field. Regions consisting of sensor nodes with acoustic signal strength above a threshold represent targets in the field. Count of these regions is used to determine the total number of targets in the field.

Query resolution:

Localized approach: The query node broadcasts the query over the wireless channel. When a sensor node receives the query for the first time it performs the following steps: (i) it stores the id of the neighbor from which it received the query as its parent, (ii) it rebroadcasts the query in the network, and (iii) if it satisfies the queried feature, it unicasts a response message with its id and feature value to the parent. When a sensor node receives a response message, it unicasts it to its parent. The above steps are repeated until the query is received by all the sensor nodes and the response messages are received by the query node. The query node computes locally upon the gathered data to identify and count the number of feature regions in the network.

$\Omega(\frac{\sqrt{n}}{r})$ timesteps are required for routing the query to all the sensor nodes and the overall energy dissipation is $\Omega(n)$ units. In the worst case scenario, all sensor nodes satisfy the queried feature. n response messages are routed to the query node, and each message traverses $\Omega(\frac{\sqrt{n}}{r})$ hops on the average. The responses cannot be aggregated at intermediate sensor nodes. This is because the sensor nodes cannot locally determine whether two feature nodes (that are not in adjacent cells) belong to the same feature region. Since the query node can receive only one message in one timestep, it takes $\Omega(n + \frac{\sqrt{n}}{r})$ timesteps to route the response messages. Thus, the query is resolved in $\Omega(n)$ timesteps dissipating $\Omega(n \cdot \frac{\sqrt{n}}{r})$ units of energy. The query node must have $\Omega(n \log n)$ bits of memory for storing the gathered data and sufficient processing capabilities for computing the result (region identification and labeling involves non-trivial computation). Since any sensor node can be a query node, all sensor nodes must meet the above resource requirements.

(b) *Forest of trees:* As in the previous case, the query is broadcast to all the sensor nodes in the network, but only the root nodes satisfying the queried feature respond to the query. The query node counts the number of responses received to compute the result. Let n_r denote the number of root nodes that satisfy the queried feature. Using sim-

ilar analysis as before, the query is resolved in response $\Omega(n_r + \frac{\sqrt{n}}{r})$ timesteps, dissipating $\Omega(n_r \cdot \frac{\sqrt{n}}{r})$ units of energy. In the worst case scenario, $n_r = n$ (all sensor nodes are roots). The query is resolved in $\Omega(n)$ timesteps with overall energy dissipation of $\Omega(n \cdot \frac{\sqrt{n}}{r})$ units. This approach is more scalable than the centralized approach as only a simple add operation is performed at the query node, and the memory requirements are reduced to $O(\log n)$ bits.

(c) *Topographic map:* The query is routed to the root node. The root stores the total number of feature regions for each feature in the network. It routes the total number of regions for the specified feature to the query node. This is accomplished in $O(\frac{\sqrt{n}}{r})$ timesteps with $O(\frac{\sqrt{n}}{r})$ units of energy dissipation. Only $O(\log n)$ bits of memory are required at the sensor nodes.

Both the topographic map approach and the forest of trees approach assume prior identification and labeling of feature regions in the network. We assume that the features in the network do not change very frequently, and thus, the overheads involved in region identification and labeling are amortized over time as more and more queries are executed in the system. We do not include these costs in the analysis, but discuss them in our simulations (See section 5).

2. Region-specific query: How many sensor nodes in feature region l have feature value above a threshold?

Application scenario: Consider a network consisting of sensor nodes whose feature values represent their remaining energy. The end user is interested in implementing a new task in the network. For the purpose of load balancing he desires to implement the task in the feature region with the largest total energy, provided that the region has at least ten sensor nodes with energy above a threshold. A global aggregate query is executed to identify the region with the largest sum of feature values. The user wants to count the number of sensor nodes in this region with energy above the threshold.

Query resolution:

(a) *Localized approach:* This approach is not applicable. Since no knowledge of topographic regions is assumed, the query cannot be routed to any specific feature region.

(b) *Forest of trees:* Let us assume that the regions are identified by the location of their root nodes. The query is routed to the root node of the specified region. The root node broadcasts the query message

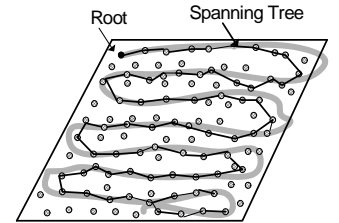


Figure 4. Snake-like region

to its children on the spanning tree in the region. The children forward the query to their own children. The above

step is repeated till the query reaches all the leaf nodes in the region tree. The leaf nodes set variable $v = 1$ if their feature value is above the threshold and $v = 0$ otherwise. They transmit the variable to their parent node on the spanning tree. Each parent node waits to receive the responses from all its children. It then sums up all the values, increments the sum by one if its own feature value is above the threshold, and sends a response to its parent on the tree. This step is repeated until the response reaches the root node, which routes the response to the query node. Let n_f denote the number of feature nodes in the region and h denote the height of the region tree. $\Omega(\frac{\sqrt{n}}{r} + h)$ timesteps are required for resolving the query with $\Omega(\frac{\sqrt{n}}{r} + n_f)$ units of energy dissipation. In the worst case scenario, $n_f = n/2$ and $h = \frac{n}{r^2}$ (see Figure 4). The query is resolved in $\Omega(\frac{n}{r^2})$ timesteps, dissipating $\Omega(n)$ units of energy.

(c) *Topographic map*: The region is labeled by the minimum id node in the region. The query is routed to this node, which unicasts the query to its clusterhead. The clusterhead routes the query to the parent of the region. The above steps take $O(\frac{\sqrt{n}}{r})$ timesteps and dissipate $O(\frac{\sqrt{n}}{r})$ units of energy. The parent routes the query to the four block leaders of the four sub blocks that constitute the larger block. The block leaders recursively repeat the above step until the query is routed to the clusterheads. Let the parent block consist of $m \times m$ clusters. Query propagation requires $k = \log m$ iterations of the above recursive step. At iteration $1 \leq i \leq k$, each of the 4^i block leaders of blocks consisting of $m/2^i \times m/2^i$ clusters route the query to the four lower-level block leaders which are at hop distance of $O(\frac{m}{2^i})$ hops. This takes $O(\frac{m}{2^i})$ timesteps and dissipates $O(m \times 2^i)$ units of energy. Summing over all k iterations, we obtain the total time and energy for routing the query to the clusterheads. Total time taken is $\sum_{i=1}^{\log m} O(m/2^i) = O(m)$, and energy dissipation is $\sum_{i=1}^{\log m} O(m \cdot 2^i) = O(m^2)$ units. The clusterheads count the feature nodes in the cluster that belong to the specified region and have feature value above the threshold. They route the count to their block leader. Each block leader waits to receive the count from the block leaders of its four sub blocks, sums up the responses, and forwards the result to the higher-level block leader. This is repeated until the count reaches the parent, which routes the result to the query node. Computation of the count requires $O(m)$ timesteps and dissipates $O(m^2)$ units of energy (using similar analysis as the query broadcast). In the worst case scenario, $m = \frac{\sqrt{n}}{s} = O(\frac{\sqrt{n}}{r})$, and the query is resolved in $O(\frac{\sqrt{n}}{r})$ timesteps, dissipating $O(\frac{n}{r^2})$ units of energy.

5. Simulation Results

An irrigation control sensor system was deployed over a 30 acre test plot in Palmdale, CA [8]. Figure 5 illustrates

the spatial distribution of hydraulic conductivity in soil at the depth of 3m, as measured by the above system. The different colors in the figure were used to define the set of features used as input for our simulations.

We simulated a system consisting of 1024 sensor nodes, uniformly distributed over a two-dimensional terrain of size 320m \times 320m. The radio range of the sensor nodes was fixed to 10.876m. The communication in the network

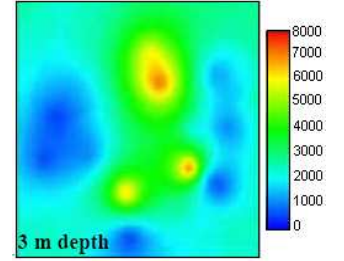


Figure 5. Hydraulic conductivity

was simulated using GloMoSim, the Global Mobile Information Systems Simulation Library [5]. Radio specifications of the mote nodes [2] were used for configuring the radios for our simulations. The two ray model was used for modeling propagation path loss in the network. Channel contention was resolved using the 802.11 protocol, and the GPSR [9] protocol was used for routing messages in the network. The sensor nodes were not time synchronized. Execution time was measured using the GLoMoSim clock, and energy dissipation was defined as the total number of messages transmitted and received in the network.

Our simulation results are illustrated in Figure 6. The y -axis of the graphs represent the (time and energy) communication costs involved in executing the following query in the network: *Count the total number of feature regions satisfying feature f in the network*. The first eight points on the x -axis represents the various feature regions over which the query was executed. The ninth point on the x -axis represents the averaged cost for executing the query over all eight features, and the tenth point represents the query: *Count all the feature regions in the network*.

Both the the *forest of trees* approach and the *topographic map* approach assume some prior state in the network. The forest of trees approach assumes that the feature regions have been identified and labeled as discussed in [10], and the topographic map approach assumes prior construction of the topographic map as described in [14]. The communication costs illustrated in the graphs for the above two approaches represent both the query execution costs and the overheads for computing the additional state. Since the *localized* approach assumes no prior state, the graphs represent only the query resolution costs for this approach.

The first row of graphs represent the costs for executing a single query in the network. For the query involving count of all feature regions in the network, the localized approach performs the worst as it involves all-to-one communication in the network. For other cases, the localized approach has the best time and energy performance on the average. This is because unlike the other two approaches, the localized

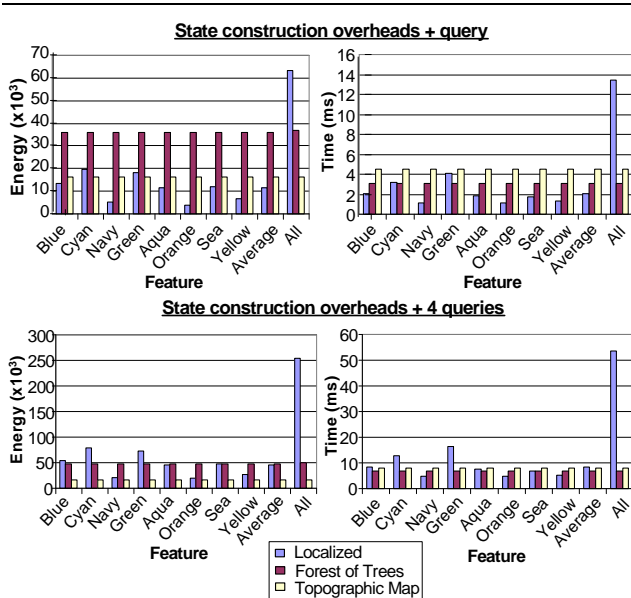


Figure 6. Simulation Results

approach does not have any state computation overheads. The second row of the graphs illustrate the costs of executing four queries in the network. The costs for the localized approach increase four times, but the increase is significantly lower for the other two approaches. This is because the state computation overheads are incurred only once and query execution costs are low for these approaches.

The topographic map approach dissipates significantly lesser energy than the forest of trees approach. It begins to show better energy performance than the localized approach after four queries. The forest of trees approach starts outperforming the localized approach in overall energy dissipation after ten queries. The centralized approach shows the best time performance for small sized regions (consisting of fewer than 20 sensor nodes), but its performance degrades for larger regions. The forest of trees approach shows marginally better time performance than the topographic map approach. Both approaches take the same time for query execution, but the state computation takes longer for the topographic map approach. This is because the feature regions in Figure 5 have small diameters (see [14] for details).

6. Concluding Remarks

In this paper, we exploited a “stateful” approach towards resolving topographic queries in sensor networks based on prior construction and maintenance of topographic maps in the system. We discussed an energy efficient algorithm for distributed construction of topographic maps in [14]. Some of our future work will focus on developing effi-

cient techniques for low-cost maintenance and update of these maps. We will investigate the overheads involved in state-replication for the purpose of enhancing the robustness of our algorithms. We will also explore resolution of other “more complex” queries using the topographic map.

References

- [1] J. Burrell, T. Brooke, and T. Beckwith. Vineyard computing: sensor networks in agricultural production. In *Pervasive Computing Magazine, IEEE*, pages 38–45, March 2004.
- [2] The CrossBow Corporation, <http://www.xbow.com>.
- [3] Q. Fang, F. Zhao, and L. Guibas. Counting targets: Building and managing aggregates in wireless sensor networks. Technical Report P2002-10298, PARC, June 2002.
- [4] D. Ganesan, D. Estrin, and J. Heidemann. DIMENSIONS: Why do we need a new data handling architecture for sensor networks? In *International Workshop on Hot Topics in Networks*, October 2002.
- [5] The Global Mobile Information Systems Simulation Library, <http://pcl.cs.ucla.edu/projects/gloimosim/>.
- [6] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Towards sophisticated sensing with queries. In *International Workshop on Information Processing in Sensor Networks*, April 2003.
- [7] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *International Conference on Mobile Computing and Networking*, August 2000.
- [8] N. B. Y. P. J. E. H. D. E. J. Kim, J. Saez and T. Harmon. Network sensing of nitrate in support of groundwater quality protection. Annual Research Review Poster, Center for Embedded Networks Sensing (CENS), 2004.
- [9] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. ACM/IEEE MobiCom*, August 2000.
- [10] B. Krishnamachari and S. S. Iyengar. Efficient and fault-tolerant feature extraction in sensor networks. In *International Workshop on Information Processing in Sensor Networks*, April 2003.
- [11] L. Lamport and N. Lynch. *Distributed Computing: Models and methods*. Formal Models and Semantics, volume B of Handbook of Theoretical computer Science, 1990.
- [12] J. Liu, M. Chu, J. Liu, J. Reich, and F. Zhao. State-centric programming for sensor-actuator network systems. In *IEEE Pervasive Computing*, 2003.
- [13] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *International Workshop on Information Processing in Sensor Networks*, April 2003.
- [14] M. Singh, A. Bakshi, and V. K. Prasanna. Constructing topographic maps in networked sensor systems. In *International Workshop on Algorithms for Wireless and Mobile Networks*, August 2004.
- [15] The klamath river fbw studies, http://www.krisweb.com/krisweb_kt/klamfbw.htm.
- [16] The sensorweb project, <http://basics.eecs.berkeley.edu/sensorwebs>.