A Location Based Aggregation Algorithm for Selective Aggregate Queries in Sensor Networks

Muhammad Umer Lars Kulik Egemen Tanin

National ICT Australia
Department of Computer Science and Software Engineering
University of Melbourne, Victoria 3010, Australia
{mumer,lars,egemen}@csse.unimelb.edu.au

ABSTRACT

In-network data aggregation algorithms are based on the premise that the energy requirements for data collection in sensor networks (SNs) can be significantly reduced by aggregating and collecting individual sensor readings over an efficient data collection path. In this work, we focus on selective aggregate queries, i.e., queries that aggregate data only from a subset of all network nodes. The task of optimal data collection in such queries is an instance of the NP-hard minimal Steiner tree problem. We present an aggregation algorithm, called Pocket Driven Trajectories (PDT) that optimizes the data collection path by approximating the global Steiner tree with a minimal overhead using purely local spatial knowledge. We show that selective aggregate queries can lead to various node participation scenarios characterized by spatial factors such as the distribution of participating nodes over the network, i.e., clustered vs. dispersed, location of clusters, inter-cluster dispersion, location of the sink with respect to the participating nodes, and location and size of communication holes. Our experiments compare the performance of well-known in-network aggregation algorithms against PDT in partial node participation scenarios. A globally approximated minimal Steiner tree serves as a benchmark for all of the aggregation algorithms. We show that PDT (a) leads to considerable gains in selective aggregate queries and (b) provides a close approximation of the minimal Steiner tree.

1. INTRODUCTION

Data collection algorithms for sensor networks (SNs) exploit the fact that a sensor node consumes less energy for information processing than for communication. For aggregation operations, decomposing the aggregate operator and partially aggregating information at the node level such as computing the sum or the average of sensor readings reduces the energy consumption by reducing the amount of communication: instead of transmitting the packets of each individual node separately, a node first aggregates the incoming packets of the nodes in communication range and then communicates the aggregated information to the next node in the collection path.

In classical database management systems, query predicates limit the number of tuples that form the output relation. Cougar and TinyDB sensor database models view the SN as an ever growing relations of tuples that are distributed across the sensor nodes [12, 22]. The query predicates in these models thus limit the set of sensor nodes that contribute to the answer of a query. We refer to the sensor nodes that relay their readings to a query as the *participating nodes*. An example *selective query* is "SELECT the humidity readings FROM all sensors WHERE the temperature is below 5° for a DURATION of 48 hours at EVERY hour". All nodes fulfilling the WHERE clause form the participating node set for this query.

Major in-network aggregation schemes do not take an explicit position on the issue of query selectivity and it is implicitly assumed that for each query *all* nodes respond to an aggregation operation. It can be shown that the classical tree-based and multipath-based techniques [10, 14] generate close to an optimal number of messages for aggregation operations for a total participation scenario in a lossless SN. In this paper, we show that major in-network aggregation schemes do not remain optimal when selectivity is introduced into the aggregate queries.

For optimizing the selective aggregate queries, we perceive two main directions in SN query processing: (a) preventing that a query is sent to nodes that do not fall into the scope of that query and, therefore, do need not to be aware of it, and (b) minimizing the number of non-participating nodes in the collection path. An example for the first category, (a), is the concept of semantic routing trees [11] where optimization between queries is the main focus. Our work falls under the second category, (b).

Since communication is the most expensive operation in sensor networks [17], the energy expenditure of an in-network aggregation algorithm can be approximated as a function of

the total number of data transmissions in the network. If it is possible for a node to aggregate all incoming messages in one outgoing message, e.g., as in calculation of an AVERAGE, the energy expenditure can be further approximated as a function of total nodes involved in data aggregation and forwarding process.

We base our work on the premise that a participating node has to transmit at least one message per sampling period of an aggregate query and hence it is best to use this node and clusters of such nodes as intermediate data aggregation and forwarding points. Thus, we optimize the energy efficiency of in-network aggregation by minimizing the number of nodes that relay a message.

The main contribution of this work is a strategy that minimizes the number of nodes used in data aggregation by discovering constrained regions to grow sub-aggregation-trees and combining these trees in an efficient manner to relay the final result to the sink. This strategy is in contrast to earlier tree-based approaches, such as TAG, where the aggregation tree is created either in a random manner or using local greedy parent selection policies [12].

Our algorithm, called Pocket Driven Trajectories (PDT) is based on the insight that spatial correlation in sensor values coupled with query selectivity give rise to a set of participating nodes formed by one or more geographically clustered sets. We refer to these geographical clusters as *pockets*. The PDT algorithm first discovers the set of pockets for a given query and then aligns the aggregation tree to the spatially optimal path connecting these pockets. This path maximizes the use of participating nodes in the aggregation tree and conversely minimizes the number of non-participating nodes. The PDT algorithm is best suited to the selective aggregate queries that regularly collect data from a relatively consistent set of nodes over time. The initial set-up cost in such queries can be amortized over the query lifetime.

The task to minimize the number of non-participating nodes in the aggregation can be modelled as a Steiner tree problem, which is known to be NP hard [15]. The number of nodes used in the globally approximated Steiner tree for a given set of participating nodes can be seen as a benchmark for in-network aggregation schemes. There are already a number of approximation algorithms to compute an approximation of the Steiner tree efficiently [8, 15, 20]. These approximation algorithms, however, require global knowledge of the communication graph. This knowledge is often not available in a SN. Thus, techniques which do not require the use of global knowledge are needed. The PDT algorithm can be seen as an approximation to the minimal Steiner tree that is solely based on local information. Our experiments show that for selective aggregate queries PDT gives better performance results than major in-network aggregation schemes and is close to a globally approximated minimal Steiner tree.

The spatial impact of query selectivity leads to a number of possible node participation scenarios. Factors such as inter and intra-pocket dispersion, location of the sink, and presence of communication holes effect the performance of the aggregation algorithm. An important contribution of this paper is the performance testing of PDT and other aggregation algorithms using extensive simulations in a number of node participation scenarios. We formulate multiple parameters to capture the spatial properties of a node participation scenario and use these as a basis for our analysis.

The remainder of the paper is organized as follows. Section 2 discusses key data aggregation approaches. In Section 3, we present the PDT algorithm. Section 4 presents a detailed experimental evaluation of our approach. Section 5 summarizes our findings and discusses future work.

2. RELATED WORK

2.1 Tree-based aggregation

TinyDB [12] performs in-network query processing using a generic aggregation service called TAG (Tiny AGgregation) [10]. TAG is one of the first tree-based in-network aggregation schemes. To gather data within the SN, the sink is appointed to be the root of a tree and broadcasts its identifier and its level. All nodes that receive this message without an assigned level, determine their own level as the level in the received message incremented by one. The identifier in the message determines the parent for the nodes receiving the message. In a lossless network in which all nodes are selected by a query, the resulting collection tree is close to an optimal solution. Aggregation in TAG is implemented by a merging function, an initializer, and an evaluator, and the aggregation operator is applied at every internal node.

In order to further optimize the data aggregation process, TinyDB introduces the concept of a semantic routing tree (SRT) [11]. For a given query, only a few nodes typically have to respond. An SRT is an index over a fixed attribute A, for example, the temperature sensed by the network, where each parent maintains the range of all values of its children for the attribute A. When a query is received by a parent, it forwards the query only when at least one child satisfies the predicate. An SRT optimizes the query forwarding phase of TAG and greatly reduces the number of broadcasts required to reach the nodes selected by the query. However, the maintenance cost of SRT exceeds its benefit if it has to be maintained for varying attribute values [12]. As pointed out earlier, SRT does not focus on the data collection optimization but on the broadcast of the query.

Tree-based aggregation schemes can be extended to adapt to changing network conditions [3]: aggregation operators are pushed down in an aggregation tree and adapt to changing conditions, such as a sub-tree that generates more readings than a sibling. This approach incrementally improves on the existing scheme. In our work, we build an aggregation scheme, after retrieving the initial readings from the network, that specifically suits the conditions and readings of the network for a participation scenario.

2.2 Multi-path aggregation

The main disadvantage of tree-based aggregation has been pointed out by recent papers on multi-path aggregation algorithms [2, 14]: trees are susceptible to link and node failures

in a SN. If a link or node fails that is close to the sink, the aggregated information of an entire sub-tree might be lost. Multi-path aggregation exploits the benefits of the wireless broadcast advantage that all nodes in communication range can hear a message and propagate the aggregates toward the sink using multiple routes. Multi-path aggregation becomes, as a consequence, more robust for node failures or communication losses. In return, a multi-path aggregation algorithm has to deal with redundancy and deviations in data aggregation [5].

Currently, for each aggregation operator, finding a duplicate-insensitive algorithm that guarantees a desired accuracy is the key challenge for multi-path routing algorithms. The obvious approach to address this challenge would be to include control information with each aggregated message. The control information contains meta-information such as the node identifier of the node that participated in the creation of this aggregate. This meta-information can be used by each forwarding node to suppress duplication. Such an approach would have the same accuracy as an aggregation algorithm using a tree. The limited storage and processing capabilities of sensor nodes, however, render such a scheme impractical for large SNs. Thus, all multi-path schemes integrate much cheaper probabilistic Order and Duplicate Insensitive (ODI) methods of the sketch theory [5, 14]. Thus, the major focus of current works in multi-path aggregation is on development of better ODI algorithms to reduce the approximation error.

In [14] multi-path aggregation algorithms are seen as energy efficient as tree-based ones. This is because of the fact that each node has to transmit a message once as in any tree-based aggregation algorithm. We show that for selective queries, however, the cost of multi-path aggregation can be significantly higher than other schemes. For selective queries this observation demands a localized use of multi-path aggregation instead of applying the method to the entire network. For example, the approach presented in [13], as a hybrid aggregation scheme for combining the benefits of the two major aggregation schemes, can form a more efficient option than pure multi-path aggregation schemes. In this approach, a multi-path-based aggregation scheme is preferred to a simple tree-based aggregation scheme when the in-network aggregation operator is close to the sink; for deeper levels of aggregation tree, the operators work as if they are on a TAG-like aggregation tree because the loss of a sensor at deeper levels only marginally effects the final result.

2.3 Clustered aggregation

Clustered in-network aggregation exploits the spatial correlation of sensor readings to preserve energy [16, 21, 23]. Spatial correlation in sensed data refers to the fact that sensor readings in close proximity are typically similar. Spatial correlation is a frequent phenomenon, in particular for attributes such as temperature or humidity [7]. If a selective query has to retrieve an aggregate such as the average temperature in a certain area, then nearby nodes typically have similar readings and are geographically clustered. Hence, only one node needs to respond to an aggregate query from a cluster [21, 23] as in the Clustered AGgregation (CAG) and the Geographic Adaptive Fidelity (GAF) approaches. However, clustered in-network aggregation has a disadvantage that, e.g., in CAG, the reported results can deviate from the real sensor readings. In static clustering [16] the network is statically partitioned into grid cells. For each grid cell one node is appointed as a cluster head that acts as a gateway but every node that has to respond to a query still reports its readings. In each cell, data is routed via a local tree, aggregated at the local gateway and then communicated to the sink.

2.4 Other aggregation approaches

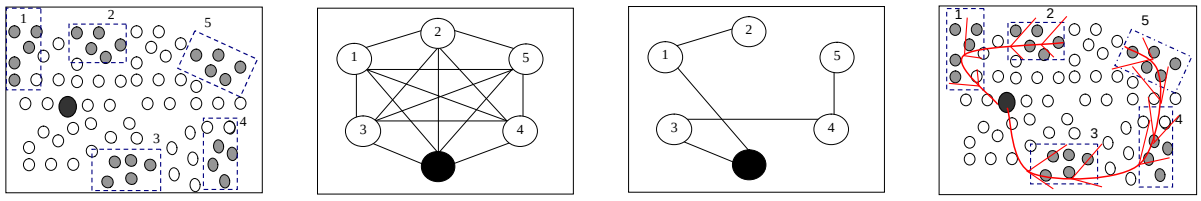
Recently, data collection schemes are characterized as spatial or temporal suppression-based techniques in [19]. Spatial suppression refers to the approaches such as clustered aggregation and model-based suppression [4] that reduce redundant transmissions by exploiting the spatial correlation of sensor nodes. Temporal suppression refers to policies that, regardless of query polling frequency, allow the retransmission from a node only if its value is changed from last transmission. In [19] a hybrid spatio-temporal suppression scheme is introduced that prevents transmission from nodes that qualify for either form of suppression. Suppression is an orthogonal data flow optimization method to our direction and can be used in tandem with our work.

In [9] the optimal data aggregation in SNs is identified as a Steiner tree problem. Authors propose a data collection scheme based on a global Steiner tree approximation [20]. The main disadvantage of their approach is the requirement that each sensor node must have global knowledge in terms of network connectivity and minimum-hop routes. In a SN comprising of devices with only basic capabilities, the maintenance of a network scale graph at each node is not practical. Moreover, the existence of such a graph at each node regardless of the query frequency and selectivity may result in unnecessary storage and communication overheads. In our work, we use entirely local information that is easily available to any node in the network. Moreover the PDT is query specific so no permanent information has to be kept or maintained.

3. A LOCATION BASED AGGREGATION ALGORITHM

A data aggregation strategy may not be able to confine the aggregation path only to the participating nodes but has to include some non-participating nodes due to the limited communication range of sensor nodes. In this context, the problem to collect aggregates back to a sink by involving minimum number of non-participating nodes can be seen as an application of the minimum Steiner tree problem: given a graph $G = (V, E)$ and a set of terminal nodes $T \subset V$, we seek a minimum cost spanning tree that includes all terminal nodes [18]. In our case, the terminal nodes are the participating nodes. The minimum Steiner tree problem is known to be NP-hard and has been widely discussed in the literature [15].

In order to compare aggregation schemes with the optimal



(a) A selective participation scenario with five pockets. Black circle represents the sink.

(b) A complete graph as an abstract representation of the scenario

(c) MST'

(d) MST' as an aggregation path

Figure 1: The computation of the PDT for a selective participation scenario

aggregation tree, the minimum Steiner tree, we outline a popular global Steiner tree approximation algorithm developed by Kou, Markowsky, and Berman [8] (henceforth referred to as KMB). The KMB algorithm allows us to compute a Steiner tree that has been shown to achieve a mean efficiency that is not worse than 5% compared to the optimal Steiner tree [6, 15]. For a graph G and a set of terminal nodes T, the KMB algorithm first computes a complete distance graph $G' = (V', E')$ for G such that $V' = T$ and the weight of each edge e' in E' is the cost of the minimum cost path between the nodes of e' in G. Then, the algorithm computes a minimum spanning tree MST' of G', translates MST' to G by substituting every edge of MST' with the corresponding path in G, and finally removes any possible cycles resulting from the translation.

The KMB algorithm is not directly applicable to SNs because, the algorithm requires global knowledge of the node connectivity for any node in the graph. Therefore, we develop a *localized* aggregation scheme, called PDT (pocket driven trajectories), that approximates the minimal Steiner tree for a selective query. The experiments in Section 4 show that the resulting aggregation tree is comparable to KMB's global approximation.

3.1 Algorithm overview

We assume that a SN with n nodes is represented by a connected unit disk graph $G = (V, E)$, where V is the set of sensor nodes and E the set of communication links. Each query Q issued by a sink S selects a subset T of V. The PDT algorithm works as follows:

1. the sink broadcasts the query Q and establishes a tree as in TAG;
2. during the first epoch the sink discovers a set of pockets $P = \{p_1 \dots p_k\}$ that partitions the set T;
3. the sink computes a complete graph $G' = (V', E')$, where $V' = P \cup \{S\}$ and each edge weight is the Euclidean distance in the SN;
4. the sink computes the minimum spanning tree MST' of G';
5. the sink establishes an aggregation tree aligned to MST';
6. the sink monitors and realigns the trajectory if the pockets change over time.

One of the key steps in PDT algorithm is the localized discovery of pockets by the sink. A pocket is a cluster of nodes selected by a query that are proximal, i.e., within a certain distance. Due to spatial correlation, pockets are common while sensing physical phenomena. We refer to the time between the two sampling operations of a query as an epoch. In the following, we describe a novel pocket discovery method, *location aggregation*, that computes pockets with minimal overhead.

Figure 1 illustrates possible pockets that are selected by a query issued at a sink. The sink broadcasts the query and all sensor nodes build a random query tree. In the first epoch, participating leaf nodes start the aggregation phase by sending the requested sensor readings to their parent nodes and by piggybacking their location information as Euclidean coordinates. Parent nodes recursively apply the query and location aggregation operators and forward the partial aggregates to their parent nodes. The query aggregation operation proceeds as in classical schemes. The key step is the location aggregation, performed in parallel to data aggregation. The location aggregation operator merges locations to an enclosing rectangle if the children nodes are proximal leaf nodes, and forwards the non-proximal nodes as singletons. If a participating node receives a rectangle, it merges the rectangle with its own position into a larger rectangle, if its position is proximal, and otherwise simply forwards the existing rectangle and singleton locations with its own location. Since the location information is piggybacked with the desired data, we expect that the location aggregation incurs a small overhead in terms of communication costs. Moreover, the successive merging of close pockets keep the volume of information small.

At the end of first epoch, the sink receives the queried aggregate and after applying location aggregation operation discovers the pockets $(p_1 \dots p_k)$ selected by the query. The sink then computes a complete graph G' as explained in the overview of the PDT algorithm (see Figure 1(a), 1(b)). The algorithm then creates a minimum spanning tree MST' for G' at the sink. MST' is a pocket driven trajectory that optimizes aggregation for the specific pocket layout (Fig-

ure 1(c)). The PDT information can be encoded as a series of locations, each corresponding to either a sink location or the center point of one of the pocket rectangles. During the next epoch, the sink establishes the PDT by broadcasting the PDT information to all its direct children. All participating nodes that receive the PDT information packet join the trajectory by reassigning their parents to that node from which they receive the PDT information. Non-participating nodes decide to join the trajectory depending upon their distance to the trajectory and only nodes that join the trajectory forward the information packet. The successive forwarding and parent switching leads to a new aggregation tree aligned with the PDT (Figure 1(d)). This new tree is afterwards used for data forwarding and aggregation. The initial TAG tree is still maintained because in future epochs previously non-participating nodes may participate. Future participating nodes might have never heard the PDT information and thus have to use the original tree for data aggregation.

3.2 Definitions

In this section we formalize the concepts of the PDT algorithm.

3.2.1 Location information

The location operator in the PDT algorithm recursively aggregates node locations. Locations that are proximal are aggregated into rectangles and non-proximal node locations are simply passed as atomic locations. More precisely, an atomic location l is a point location that cannot be aggregated by a location aggregation operator, whereas an aggregated location a is a result of such an operation. We define the location information at a node v as a set $\lambda_v = \{l_1, \ldots, l_k, a_1, \ldots, a_m\}$, i.e., a set of atomic and aggregated locations. The aggregated location a for a single atomic value l is $\{l\}$, i.e, if v is a leaf node, then $\lambda_v = \{l_v\}$. Atomic locations are represented as Euclidean coordinates and aggregated locations are minimum bounding rectangles enclosing proximal node locations.

3.2.2 Location aggregation

The aggregation operator takes in the location information of a node (the set λ_v), computes the enclosing rectangle of all atomic locations within a certain threshold (set to the communication radius), and merges the computed set with already aggregated locations reported by its children. Formally, the location aggregation operator $\mathcal{L}$ for the location information set λ_v of node v is defined as follows:

$$\mathcal{L}(\lambda_v) = \mathcal{M}\Big(\mathcal{E}\big(l_1, \ldots, l_k\big) \cup \mathcal{M}\big(a_1, \ldots, a_m\big)\Big),$$

where $\mathcal{E}$ is defined as

$$\mathcal{E} : (l_1, \ldots, l_k) \mapsto \lambda_v^{\mathcal{E}},$$

i.e., $\mathcal{E}$ maps the set of atomic locations to a location information set that consists of minimum bounding rectangles of proximal atomic locations and the non-proximal atomic locations. Nodes are proximal if their distance is below a certain threshold value ε. Similarly, $\mathcal{M}$ is defined as

$$\mathcal{M}(a_1, \ldots, a_m) \mapsto \lambda_v^{\mathcal{M}},$$

i.e., $\mathcal{M}$ merges aggregated locations with each other and with atomic locations. In order to ensure that the merged locations remain connected, locations are only merged when the distance between the farthest points on the closest sides of the corresponding rectangles are proximal. An atomic location is considered as a degenerate rectangle.

3.2.3 Computation of the PDT

The PDT $\mathcal{P}$ for a sink S is defined as a set of edges $\mathcal{P}(S) = \{(a_{i_1}, a_{i_2}), \ldots, (a_{i_{n-1}}, a_{i_n})\}$ that is computed as

$$\mathcal{P}(S) = MST(\text{Clique}(\mathcal{L}(\lambda_S))),$$

where Clique creates a complete graph G' of the final location information set $\mathcal{L}(\lambda_S)$. Here, $G' = \{V', E'\}$ such that $V' = \{p_1, \ldots, p_n, S\}$, where $p_1 \ldots p_n$ are the center points of each location aggregate in $\mathcal{L}(\lambda_S)$. The edge weights are the Euclidean distances between points in V'.

Furthermore, $MST = \{(a_{i_1}, a_{i_2}), \ldots, (a_{i_{n-1}}, a_{i_n})\}$ such that each tuple (a_j, a_k) is an edge in the minimum spanning tree of G', where (a_j, a_k) are corresponding aggregates for center points (p_j, p_k) in G'.

3.2.4 PDT join decision

The PDT join decision, $\mathcal{J}$ is a Boolean operator that determines whether or not a node should join the PDT. We use the following greedy criterion to define the join decision for a node v at location l_v:

$$\mathcal{J}(\mathcal{P}(S)) = \begin{cases} 1, & \begin{cases} d(l_v, (a_i, a_j)) < \varepsilon, \text{if } (a_i, a_j) \text{ is a link in } \mathcal{P}(S) \\ \text{or if } v \text{ is a participating node} \end{cases} \\ 0, & \text{otherwise} \end{cases}$$

where d is the distance of v and the line segment connecting the center points of the aggregates (pockets) a_i and a_j.

Algorithm 1 outlines the formal specification of the PDT setup algorithm.

3.3 Shortcomings and overheads

A query in a SN can consist of a large number of epochs. Even if the change per epoch is small, the node participation can change significantly during the lifetime of a query. The PDT algorithm is ideal if the change per epoch is relatively small so that the pockets do not change significantly in every step. Under those conditions, the PDT has to be realigned only a few times during the lifetime of a query. However, currently the PDT algorithm does not adapt to change within a query. We leave the investigation of more dynamic scenarios to future work.

The increase in packet size due to the aggregation of location information increases the communication overhead. The location information consists of aggregated pocket information and a list of atomic locations. Due to the spatial correlation of physical phenomena, the number of atomic locations is typically small and singletons mostly occur at the leaf level of the aggregation tree. The singletons are almost

Algorithm 1: PDT Setup for Node v

```
epoch_v ←∝
while epoch_v is not expired do
    if query broadcast is received then
        parent_v ← broadcast_sender
        Reset epoch_v
        Broadcast query packet
    if data packet from child c is received then
        λ_v ← λ_v ∪ λ_c
        agglist_v ← agglist_v ∪ aggdata_c
        isparent_v ← TRUE
    if PDT information packet (P) is received then
        if J(P) = 1 then
            parent_v ← PDT_sender
            Reset epoch_v
            Forward PDT information packet
if (v is participating node) OR (isparent_v = TRUE) then
    if v is participating node then
        λ_v ← λ_v ∪ {l_v}
    λ_v ← L(λ_v)
    aggdata_v ← aggregate(agglist_v)
    if v is not sink then
        Send λ_v, aggdata_v to parent_v
    else
        clique_v ← Clique(λ_v)
        P ← MST(clique_v)
        Broadcast PDT information (P)
        Forward aggregate to the application
```

completely merged at the lower levels of the tree into pockets that traverse for the remainder of the tree in a compact form as rectangles. Our experiments show that the location information aggregation only slightly increases the communication messages.

The announcement of the PDT is the other major overhead of the PDT algorithm. The number of extra messages generated during PDT information broadcast phase is equal to the number of nodes that decide to join the PDT. The initial setup cost can be amortized over the query lifetime. However, the initial cost cannot be amortized for snapshot queries and hence we do not recommend the use of the PDT algorithm for such queries.

4. EXPERIMENTAL EVALUATION

4.1 Evaluation parameters

In this section we compare the performance of the PDT algorithm and other major in-network aggregation schemes in a variety of SN settings. We first lay out the evaluation parameters that we use to analyze the impact of spatial characteristics of selective aggregate queries.

4.1.1 Spatial selectivity index

The nature and extent of spatial clustering of participating nodes may depend upon a number of factors, such as the magnitude of spatial correlation, SN configuration, the query predicate, and so forth. We introduce an integrated measure, called spatial selectivity index, SSI, that describes the extent of spatial clustering in a SN. SSI is based on two key parameters, *node scattering* and *pocket scattering*.

The *node scattering*, NS, at time t (we will drop the argument t for the sake of simplicity) describes the distribution of pockets for a query as the ratio of the total number of pockets P relative to the number of participating nodes N:

$$\mathrm{NS} = P/N$$

The *pocket scattering*, PS, characterizes the degree of dispersion for the pocket locations. We define the *pocket centroid*, PC, of a pocket P_i as the sensor node that is closest to the average location of all nodes belonging to P_i. We then define the *global pocket centroid*, GPC, as the node that is closest to the centroid of all pocket centroids $P_1, \ldots, P_l$ partitioning the nodes selected by the query:

$$\mathrm{GPC} = \frac{1}{l} \cdot \sum_{i=1}^{l} \mathrm{PC}(P_i)$$

Let $\mathrm{HC}(v, v')$ denote the minimum number of hops for a path connecting two nodes v and v'. Then, the pocket scattering PS is defined as the average of hop counts connecting the global pocket centroid with the pocket centroids of the pockets $P_1, \ldots, P_l$:

$$\mathrm{PS} = \frac{1}{l} \cdot \sum_{i=1}^{l} \mathrm{HC}(\mathrm{PC}(P_i), \mathrm{GPC})$$

We use the hop count between two nodes as a distance measure instead of their Euclidean distance. Since deployments of SNs can exhibit holes and barriers, the hop count provides a realistic approximation of the actual communication cost. It should be noted that the use of hop count as a distance measure is purely for evaluation purposes. The PDT algorithm itself does not use the hop count measure due to the practical limitations in making such information available locally at each node.

The spatial selectivity index SSI is then defined as a measure that describes the impact of node scattering as well as of pocket scattering for a given scenario:

$$\mathrm{SSI} = \mathrm{NS} \cdot \mathrm{PS}$$

Lower SSI values characterize scenarios that are well pocketed and have a small pocket scatter, for example see Figure 3(a) and 3(b) in Section 4.3.1 that show two network deployments with different SSI values.

4.1.2 Sink centroid distance

We formalize sink position in order to analyze the affect of sink location on the performance of an aggregation algorithm. The ideal position of a sink is the GPC. The *sink centroid distance*, SCD, measures the hop count between the sink and its ideal position. If S is the position of the sink, the sink centroid distance SCD is defined as

$$\mathrm{SCD} = \mathrm{HC}(S, \mathrm{GPC})$$

4.1.3 Co-connectivity of a deployment

Although a rectangular deployment is common in simulations, e.g., without any holes that alter connectivity, in this paper, we also consider the impact of irregular deployments on aggregation algorithms. Particularly, we measure the impact of holes. To simplify our discussion, we only take the total number of holes and their normalized size into account. If $|\mathcal{R}|$ denotes the size of the deployment area $\mathcal{R}$, then the normalized size of a hole H_i is $|H_i|/|\mathcal{R}|$. The *co-connectivity measure* CC is then defined as the sum of the normalized sizes of all holes:

$$\text{CC} = \sum_i |H_i|/|\mathcal{R}|$$

We summarize the indices to measure the spatial features of an aggregation query in Table 1.

Spatial Features	*Performance Index*
How well-pocketed is a participation scenario?	Node Scatter (NS)
How scattered are the pockets?	Pocket Scatter (PS) Spatial Selectivity Index (SSI)
How far is the sink from the pockets? Where is the query issued?	Sink Centeroid Distance (SCD)
How many holes are in the deployment region?	Co-connectivity (CC)

Table 1: Simulation parameters

4.2 Simulation setup and methodology

In addition to PDT, we implement comparable aggregation algorithms discussed in Section 2. We implement all algorithms in Network Simulator-2 (NS-2) [1]. To provide a lower bound, we compute for each experiment an approximation of the optimal aggregation tree using the KMB algorithm. In our simulations, we collect the AVERAGE on a deployment of 750 nodes, placed randomly in a 75m x 75m grid. Each query collects data from a SN for 100 epochs. We utilize the NS-2 wireless communication infrastructure that simulates 914 MHz Lucent Wave LAN DSSS radio interface using the two ray ground reflection propagation model and IEEE 802.11 MAC layer (Chapters 16 & 18 of the NS-2 Manual [1]). Communication is performed using omni-directional antennas centered at each node, while the communication radius is fixed at 5m. The message payload is fixed at 72 bytes and we assume that every algorithm has the same payload for data transfers. Furthermore, we assume a lossless network with synchronized many-to-one aggregation, i.e., during in-network aggregation each internal node is perfectly synchronized with its children and after aggregation it always emits just one packet. Table 2 summarizes these parameters.

We use total data transmission (in MBs) as an indication of energy usage and hence as the basic metric of performance comparison. The amount of data transmission can be related to the energy expenditure by a simple function such as $\varepsilon = \sigma_s + \delta_s x$, where ε is the total amount of energy spent in sending a message with x bytes of content, and σ_s and δ_s represent the per-message and per-byte communication costs, respectively [19].

Deployment Area:	75m × 75m
Number of Nodes:	750
Wireless Model:	914 MHz Lucent Wave LAN DSSS radio interface
Propagation Model:	Two ray ground reflection
MAC Layer:	802.11
Communication Radius:	5m
Packet Size:	72 Bytes
Number of Epochs:	100
Algorithms:	TAG [10] Static Clustering (SC) [16] Multi-path (MP) [14] PDT

Table 2: NS-2 parameters

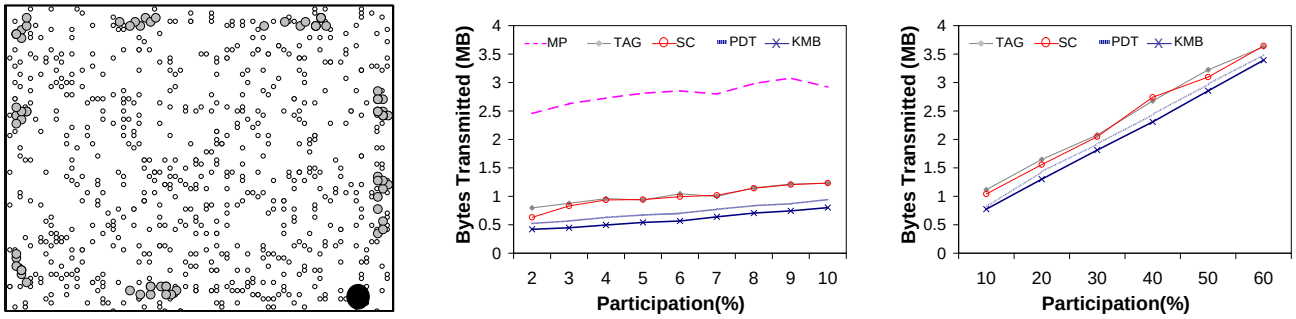
In order to systematically study the impact of varying participation levels and selective participation measures as defined in Section 4.1, we design our experiments according to the following questions:

- What is the impact of query selectivity and level of spatially correlated node participation on the performance of aggregation algorithms?
- What is the impact of the position of pockets and their dispersion in selective participation scenarios?
- What is the impact of the location of the sink on the performance of an aggregation algorithm in low node participation levels?
- What is the impact of (communication) holes on data collection?

4.3 Results

4.3.1 Impact of query selectivity

In two experiments, we investigate the impact of the query selectivity on four different aggregation schemes: PDT, MP, SC, and TAG. In the first experiment, Figure 2, we change the selectivity of an aggregate query and hence the number of nodes that participate in a query by 1% increments from 2% to 10% of the total nodes in the SN. The participating nodes are spatially clustered (see the deployment snapshot in Figure 2(a)). Figure 2(b) shows the mean value of the number of bytes transmitted by each algorithm at each participation level (the average of five runs is used to find the mean value). Figure 2(c) shows results from a similar experiment with participation levels ranging in discrete steps from 10% to 60%.

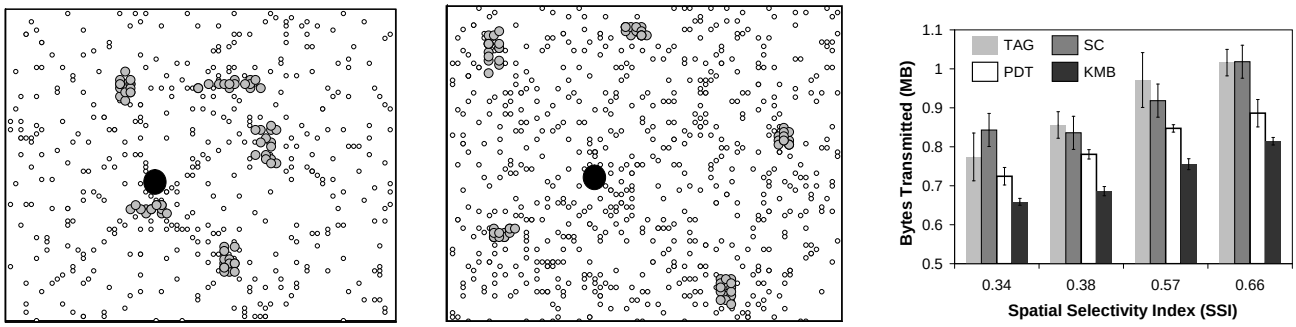


(a) A spatial configuration for partial node participation. Larger circles represent nodes that participate in a query. Black circle represents the sink.

(b) Detailed comparison of data transmissions for each algorithm in low participation scenarios (the participation level ranges between 2%–10%)

(c) The overall trend for each algorithm in terms of data transmissions (the participation level ranges between 10%–60%)

Figure 2: The performance of aggregation techniques for varying levels of partial node participation



(a) A partial node participation scenario with an SSI of 0.34. Black circle represents the sink.

(b) A partial node participation scenario with an SSI of 0.57

(c) The impact of an increasing SSI on the performance of aggregation algorithms for a 10% node participation level. Standard deviations are shown.

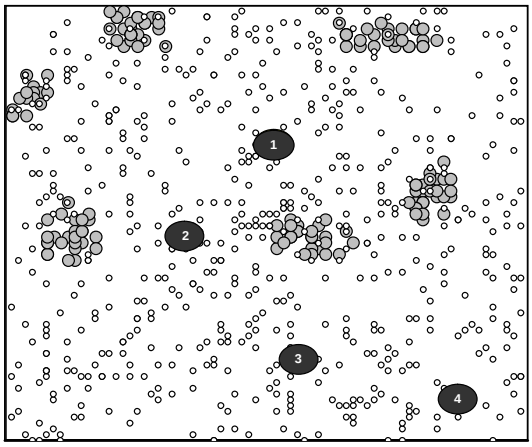
Figure 3: Various network configurations simulating an increase in the spatial selectivity index by increasing the pocket dispersion for a 10% node participation level

Figure 2(b) shows that for low node participation levels, PDT performs better than other aggregation schemes. For participation levels from 2% to 10% PDT is, on average, 41% more efficient than TAG and 37% more efficient than SC. In addition, PDT is just 21% less efficient than the approximated lower bound, where TAG and SC are 72% and 67% less efficient, respectively. This experiment also reveals that the energy consumption for MP in low participation scenarios is significantly higher than all other aggregation algorithms. MP requires at least 2.7 and 2.8 times as much data transmission as TAG and SC, respectively and 3.8 times more than PDT. Similarly, the trend in Figure 2(c) shows that PDT remains energy efficient even for high participation levels but its advantage decreases as the participation levels increase. At 10% participation, PDT requires 30% less data transmissions than TAG and 31% less than SC; however at the participation level of 60% this lead reduces to 4% and 5% for TAG and SC, respectively. The decrease in efficiency results from the fact that with the increase in node participation the benefit of spatial correlations diminishes. This effect can also be observed from the fact that at 60% participation level PDT is just 3% less efficient than the KMB lower bound. For high participation levels, MP is not shown on the figure to simplify the presentation.

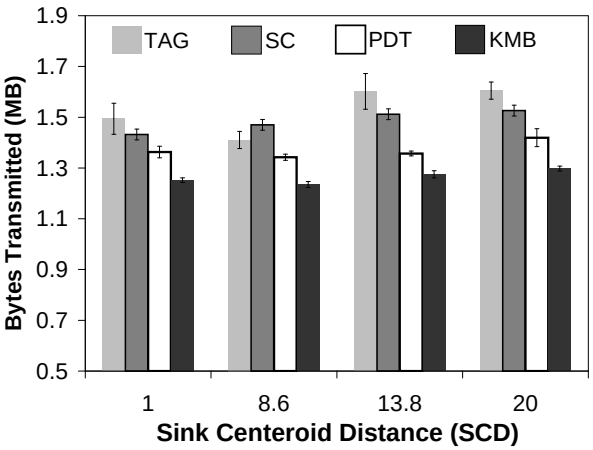
4.3.2 *Impact of varying the spatial selectivity index*

This section describes a set of experiments that assess the performance of the PDT algorithm in various spatial layouts, characterized by different SSI values (Section 4.1.1). A low SSI value represents a low dispersion scenario. We expect a better performance from PDT and other aggregation algorithms for selective queries with low SSI values. In this experiment, we achieve the effect of increasing SSI values by expanding the dispersion of pockets in the network, while the total number of pockets, the node participation level, and the sink position remains constant. Figure 3(a) and 3(b) show the deployment snapshots of two scenarios.

Figure 3(c) confirms the hypothesis that all algorithms perform better for lower SSI values (standard deviations are also shown in this chart). As the dispersion of the participating nodes increases, all algorithms have to spend more



(a) The spatial distribution of the network configuration and the various positions of the sink



(b) The performance of the aggregation algorithms for different sink positions with an increasing distance to the global centroid position for a 20% node participation level

Figure 4: A network configuration with different sink positions for a 20% node participation level

energy. For the analyzed scenarios, TAG and SC transmit up to 31% and 22% more data for the highest SSI value. PDT also generates more data and shows an increase of 22% for the highest SSI value, however it remains 15% energy efficient than both TAG and SC.

4.3.3 *Impact of the location of the sink*

In this experiment we analyze the effect of sink position on PDT and other aggregation schemes. Figure 4 shows the performance of each algorithm in a deployment where the same query is issued from different sink positions. The chart shows that, for the given deployment, different sink positions affect the overall cost only modestly: between initial and final sink position data transmission rises by just 7% for both SC and TAG, while it rises to only 4% for PDT. The average change in cost from one scenario to next is 1%, 2% and 3% for PDT, SC, and TAG respectively.

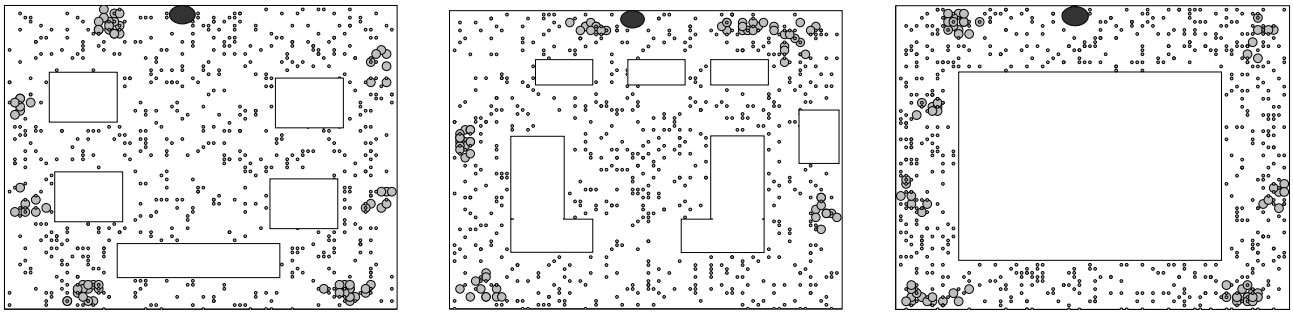
The result is not surprising for the SC and PDT algorithms. In PDT the aggregation tree is mostly determined by the pocket locations while the impact of sink location is limited to the distance between the sink and the pockets closest to it. Similarly, SC always uses fixed paths to aggregate data inside each cluster and the impact of sink location is limited to the final phase where cluster heads have to route the aggregated data to the sink. The bulk of data transmission in both cases occurs inside pockets (or clusters) and as a result the impact of the position of sink is reduced. However, it is important to note that the behavior of TAG fluctuates with the sink location. Among the simulated scenarios, the second sink position is the best for TAG. At this position, the sink is located in a way that paths to distant pockets naturally emerge from the closer pockets, resulting in an increased number of participating nodes acting as intermediate nodes in the tree. In other scenarios, the misalignment of the root of the tree, sink, with the clusters increases the cost for TAG.

4.3.4 *Spatial layout and communication holes*

Real SN deployments cannot stay fully connected in a regular grid structure although many routing and in-network aggregation algorithms are commonly tested on such basic structures. Due to constrained communication capabilities, a network might be disconnected at certain places leaving gaps that we name as communication holes. In the context of in-network aggregation, if a given network suffer from communication gaps while still remaining connected via alternate communication routes, it is of interest to understand how the presence of holes effect the performance of an aggregation algorithm.

To investigate the effect of communication holes we simulate three different SN configurations. The configurations are shown in the deployment snapshots in Figure 5(a)–Figure 5(c), where the bordered regions represent communication holes. In each of the deployments, we set up a 10% pocketed participation scenario and Figure 5(d) shows the performance of each algorithm in these spatial layouts. In TAG, we see that a collection of communication holes can affect the formation of the aggregation tree in one of two ways. Firstly, the holes might break the most direct communication paths to pockets and hence the tree has to invariably take a longer route. This effect can be observed from the high cost of TAG in the first deployment (Figure 5(a)). However, a second more interesting scenario is where the presence of holes actually reduces the communication cost by restricting the tree into a set of corridors that naturally spans the pockets. The cost of TAG is reduced by 35% between the initial and final deployment.

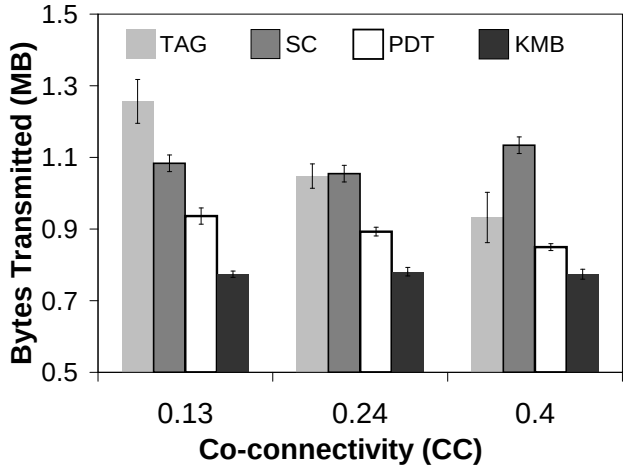
SC is rather unaffected by the presence of holes where the



(a) A network configuration with deployment co-connectivity of 0.13 (five holes relatively close to the border)

(b) A network configuration with deployment co-connectivity of 0.24 (six randomly distributed holes)

(c) A network configuration with deployment co-connectivity of 0.40 (one large hole in the middle of the network)



(d) The impact of different co-connectivities on the aggregation algorithms for a 10% node participation level

Figure 5: Various network configurations simulating deployments with different types of communication holes and a 10% node participation level

change in its cost between the initial and final deployments is just 5%. PDT also performs almost unaffected by the presence of communication holes and shows a 10% reduction in data transmissions. PDT might suffer in cases where there is a communication gap between two neighboring pockets in the trajectory.

4.4 Discussion

With extensive experiments, we presented the advantages of the location based PDT algorithm for in-network aggregation over well-known in-network aggregation algorithms in different SN settings. Our results validate the hypothesis that a variable node participation scenario affects the performance of existing algorithms. In addition, the spatial features of the scenario do also have an effect on the performance of in-network aggregation.

The performance of an in-network aggregation strategy in selective queries can be presented as a function of the number of nodes the aggregation paths utilize while aggregating data from the participating nodes. The high cost of the tree-based scheme in highly selective queries can be explained by the random strategy used in the creation of the aggregation tree where no query specific optimization is considered during the tree construction process. We observe that for low node participation levels, the tree can improve its performance when the query sink is aligned with the data pockets in a way that paths to distant pockets naturally emerge from the pockets that are close to the sink. Similarly, a tree-based strategy shows considerable improvement in performance if the communication channels in the network are constrained by holes. However, in realistic settings, such cases may be rare and may not justify building a random tree for low participation scenarios. In contrast to such special cases, PDT always identifies constrained regions to grow sub-aggregation trees and an overall collection path in an efficient manner. Since the data collection is optimized to minimize the number of non-participating nodes en-route to data sources, PDT shows an overall reduction in data transmission even in high node participation scenarios. The

experiments also show that the cost of PDT rises with increasing participation levels or decreasing spatial correlation levels, however for long running queries with many epochs, it is at least as good as the other aggregation schemes.

As an interesting result, we observe that static clustering shows comparable results to the tree-based strategy even though it is not configured as dynamically as the tree-based strategy. Similar to the tree-based strategy, we also observe that static clustering performs better if the location and size of pockets correlate with that of the statically configured clusters and the hence the data collection mechanism. Static clustering algorithm proposes to define the cluster size parameter depending upon the degree of spatial correlation in the network [16]. A major challenge in static clustering is determining the correct cluster size for queries with complex, selective, predicates.

In our experiments where the node participation levels are low, data transmission by multi-path aggregation algorithm has greater costs than all the other schemes. This is an interesting observation since in exhaustive participation scenarios multi-path achieves the same number of messages per node as a tree [13]. Although the goal of multi-path aggregation is to achieve accuracy in lossy environments rather than increasing efficiency of the aggregation strategy, the large cost of multi-path in low node participation scenarios may require adopting a hybrid method such as [13].

One important feature of the PDT algorithm is that there is a trade-off between the latency of data collection and the aggregation path data transmission costs for a given query. In order to minimize the number of non-participating nodes in the aggregation process, PDT creates paths with possibly higher latencies. Since in a sensor network saving energy is the primary concern, reducing response time may not impact many query types. Thus, the latency is not analyzed as a separate parameter with our experiments.

Summary of Results

1. PDT performs significantly better than well-known data aggregation algorithms for long running aggregation queries that select $2\% - 10\%$ of the nodes that are spatially correlated into pockets.
2. PDT shows better cost savings when the participation levels are low and the pockets are far from the sink.
3. For selective queries, the performance of all algorithms and the difference between PDT and the other aggregation algorithms depends upon the spatial selectiveness of a given query.
4. The performance of tree-based schemes in low node participation scenarios is improved in cases where the pockets are aligned in a manner that paths to pockets that are afar from the sink emerge naturally from the pockets closer to the sink. (Similarly, a tree performs better when communication channels are constrained by holes in a way that data collection paths naturally span the data pockets.)
5. In low node participation scenarios a tree and static clustering have similar costs although static clustering does not use as dynamic a strategy as the tree-based algorithm.
6. The difference in cost between aggregation strategies becomes less significant for high participation scenarios and algorithms behave almost the same for exhaustive queries. PDT does not perform worse than other algorithms even for high participation scenarios despite its overheads.
7. In low node participation scenarios, multi-path aggregation is significantly more expensive in terms of data transmission costs than any other algorithm.

5. CONCLUSIONS AND FUTURE WORK

Selective queries are required for effective monitoring of physical phenomena using SNs. In this paper, we introduce the PDT algorithm, an in-network data collection method for long running selective aggregation queries. With extensive simulations, we show that PDT is more energy efficient than other major in-network aggregation schemes under various scenarios and is close to a well-known approximation of the global optimum, i.e., the Steiner tree. PDT algorithm discovers pockets of participating nodes using purely local information and approximates a minimum Steiner tree for data collection from these pockets. We show that this can lead to significant energy savings in different node participation scenarios. We define spatial parameters to characterize a specific network deployment and present how our work compares to different well-known data aggregation algorithms under various sensor deployments using these parameters.

There are several research directions that we plan to follow as our future work. For a given query, an initial participation scenario might change over time. For example, a change in the physical conditions for the sensed phenomena can lead to changes in values being sensed by the network. For a selective query this change can have an impact on multiple fronts, i.e., by increasing or decreasing the node participation levels, by changing the distribution of participating nodes such as the emergence of new pockets or breakdown of old pockets. Thus, it is important to study a variety of trends of change and introduce new improved strategies for aggregation to continuously adapt to these changes. We plan to extend the PDT algorithm to be able to continuously adapt to change. A possible strategy in this regard is to perform the location aggregation during some sampling periods and detect the amount of deviation from the original PDT. The PDT can then be refreshed as soon as this deviation reaches to a certain threshold value.

We also plan to investigate the impact of the length of a query on the PDT algorithm in comparison to other aggregation algorithms. In addition, we plan to study the latency incurred by the PDT algorithm by increasing the data collection path while decreasing the energy consumption. Furthermore, we plan to improve our PDT algorithm by a guided query multicast phase as opposed to the query broadcast re-

ported with this paper. One idea is to combine the SRT concept from TinyDB to optimize the query forwarding phase. Finally, our work can be viewed as a method to discover restricted regions inside which a query-tree maximizes the use of the participating nodes. Once such pockets are discovered there can be many aggregation schemes that can be used in these pockets. This paper reports only the application of a random tree-based aggregation algorithm within a pocket, however, we plan to investigate the efficiency of other methods for aggregation inside a pocket including various suppression strategies.

6. REFERENCES

[1] The network simulator NS-2 documentation, http://www.isi.edu/nsnam/ns/ns-documentation.html.

[2] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The price of validity in dynamic networks. In *Proceedings of the SIGMOD*, pages 515–526, 2004.

[3] B. J. Bonfils and P. Bonnet. Adaptive and decentralized operator placement for in-network query processing. In *Proceedings of IPSN*, pages 47–62, 2003.

[4] D. Chu, A. Deshpande, J. Hellerstein, and W. Hong. Approximate data collection in sensor networks using probabilistic models. In *Proceedings of ICDE*, page 48, 2006.

[5] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of ICDE*, pages 449–460, 2004.

[6] M. Doar and I. M. Leslie. How bad is naive multicast routing? In *Proceedings of INFOCOM*, pages 82–89, 1993.

[7] H. Gupta, V. Navda, S. R. Das, and V. Chowdhary. Efficient gathering of correlated data in sensor networks. In *Proceedings of MobiHoc*, pages 402–413, 2005.

[8] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15:141–145, 1981.

[9] B. Krishnamachari, D. Estrin, and S. B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of ICDCSW*, pages 575–578, 2002.

[10] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of SIGMOD*, pages 491–502, 2003.

[12] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[13] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: efficient and robust aggregation in sensor network streams. In *Proceedings of SIGMOD*, pages 287–298, 2005.

[14] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of SenSys*, pages 250–262, 2004.

[15] C. A. S. Oliveira and P. M. Pardalos. A survey of combinatorial optimization problems in multicast routing. *Comput. Oper. Res.*, 32(8):1953–1981, 2005.

[16] S. Pattem, B. Krishnamachari, and R. Govindan. The impact of spatial correlation on routing with compression in wireless sensor networks. In *Proceedings of IPSN*, pages 28–35, 2004.

[17] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[18] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of SODA*, pages 770–779, 2000.

[19] A. Silberstein, R. Braynard, and J. Yang. Constraint chaining: on energy-efficient continuous monitoring in sensor networks. In *Proceedings of SIGMOD*, pages 157–168, 2006.

[20] H. Takahashi and A. Matsuyama. An approximate solution for the Steiner problem in graphs. *Math Japonica*, 24:573–577, 1980.

[21] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of MobiCom*, pages 70–84, 2001.

[22] Y. Yao and J. Gehrke. Query processing for sensor networks. In *Proceedings of the Conference on Innovative Data Systems*, pages 233–244, 2003.

[23] S. Yoon and C. Shahabi. Exploiting spatial correlation towards an energy efficient clustered aggregation technique (CAG). In *IEEE International Conference on Communications*, pages 82–98, 2005.