

# Location-Aware Routing for Data Aggregation in Sensor Networks<sup>1</sup>

Jonathan Beaver, Mohamed A. Sharaf,  
Alexandros Labrinidis, Panos K. Chrysanthis

Advanced Data Management Technologies Laboratory  
Department of Computer Science  
University of Pittsburgh  
Pittsburgh, PA 15260, USA  
{beaver, msharaf, labrinid, panos}@cs.pitt.edu

## ABSTRACT

In-network aggregation has been proposed as one method for reducing energy consumption in networked sensors. In this paper, we explore the idea of *influencing the construction of the routing trees for sensor networks* with the goal of reducing the size of transmitted data for networks with in-network aggregation involving Group By queries. Toward this, we propose a *group-aware network configuration* method and present two algorithms, that “cluster” along the same path sensor nodes which belong to the same group. We evaluate our proposed scheme experimentally, in the context of existing in-network aggregation schemes, with respect to energy consumption and quality of data. Overall, our routing tree construction scheme provides energy savings over existing network configuration schemes and improves quality of data in systems with imperfect quality of data such as TiNA.

## 1 INTRODUCTION

From monitoring endangered species [7, 12], to monitoring structural integrity of bridges [8], to patrolling borders, sensor networks today offer an unprecedented level of interaction with the physical environment. Within a few years, miniaturized, networked sensors have the potential to be embedded in all consumer devices, in all vehicles, or as part of continuous environmental monitoring.

Sensor nodes, such as the Berkeley MICA Mote [4] which gathers data such as light and temperature, are getting smaller, cheaper, and able to perform

---

<sup>1</sup>Supported in part by the National Science Foundation award ANI-0325353.

more complex operations, including having mini operating systems embedded in the sensor [5]. While these advances are improving the capabilities of sensor nodes, there are still many crucial problems with deploying sensor networks. Limited storage, limited network bandwidth, poor inter-node communication, limited computational ability, and limited power still persist.

One way of alleviating the problem of limited power is by employing in-network query processing instead of query processing at the base station. For example, assume a sensor network that is used to monitor the average temperature in a building. One way to implement this is to have each sensor send its temperature reading up the network to the base station, with intermediate nodes responsible for just routing packets. Another way, with in-network query processing (or aggregation), would be for each node to incorporate its own reading with the average computed so far by its children. In this way, only one packet needs to be sent per node and each intermediate node computes the new average temperature before sending information further up the network.

As the example shows, with in-network aggregation some of the computational work of the aggregation is performed within the sensor node before it sends the results out to the network. The reason why in-network aggregation reduces power consumption is that sensor power usage is dominated by transmission costs, as has been shown in [3, 6]. Therefore, being able to transmit less data (the result of the aggregation over having to forward all the packets) results in reduced energy consumption at the sensor nodes.

In this work we explore the idea of *influencing the construction of the routing trees for sensor networks* with the goal of reducing the size of transmitted data, especially with in-network aggregation. More specifically, in addition to traditional link-strength criteria, the idea is to consider the semantics of the query and the properties/attributes of the sensor nodes when configuring the sensor network and in particular building the routing tree for the aggregation. Based on this idea, we propose a *group-aware network configuration* method and developed two algorithms, called *GaNC* and *GaNCi*, that “cluster” along the same path sensor nodes that belong to the same group. The intuition of this approach is that messages along such paths will contain less groups and hence incur less energy cost in transmitting them.

We have experimentally evaluated our proposed group-aware network configuration algorithms using simulation. We have investigated the improvement

in energy for group-aware network configuration for the sensor network implementations of TAG and Cougar, which are two representative schemes for in-network aggregation. We have further considered our algorithms in conjunction with a new energy efficient scheme for in-network aggregation called TiNA (Temporal coherency-aware in-Network Aggregation). Our results show that by using group-aware network configuration we have savings in energy of up to 33% over the strongest link method and in the case of TiNA, the proposed method can help improve the quality of data it provides while further increasing energy savings.

The rest of this paper is organized as follows. Section 2 provides an overview of in-network aggregation and the TiNA scheme. Additional background in sensor network routing tree configuration is provided in Section 3. The proposed network configuration algorithms, *GaNC* and *GaNCi*, are presented in Section 4. Section 5 describes our simulation testbed, and then in Section 6 we show our experiments and results. We present related work in Section 7. We conclude in Section 8.

## 2 BACKGROUND

In this paper we propose network configuration and routing techniques to further save energy in sensor networks. Before presenting the proposed algorithms, we give a brief overview of current in-network aggregation schemes; our proposed techniques work in conjunction with all such schemes.

### 2.1 In-Network Aggregation

Directed diffusion [2, 6] is the prevailing data dissemination paradigm for sensor networks. In directed diffusion data generated by a sensor node is named using attribute-value pairs. A node requests data by sending interests for named data. Data matching the interest is then drawn towards the requesting node. Since data is self-identifying, this enables activation of application-specific caching, aggregation, and collaborative signal processing inside the network, which is collectively called *in-network processing*. Ad-hoc routing protocols (e.g., AODV[13], Information-directed Routing[9]) can be used for request and data dissemination in sensor networks. These protocols, however, are end-to-end and will not allow for in-network processing. On the contrary, in directed diffusion each sensor node is both a message source and a message sink at the same time. This enables a sensor to seize a data packet that it is forwarding on

behalf of another node, do in-network processing on this packet, if applicable, and forward the newly generated packet up the path to the requesting node.

The work on *Cougar* [1, 19] and *TinyDB* [10, 11] introduced the directed diffusion concepts in the database arena. Cougar abstracted the data generated by the sensor network as an append-only relational table. In this abstraction, an attribute in this table is either information about the sensor node (e.g., id, location) or data generated by this node (e.g., temperature, light). Cougar and TinyDB emphasize the savings provided by using in-network aggregation, which is one type of in-network processing. Sensor applications are often interested in summarized and consolidated data that are produced by aggregated queries rather than detailed data.

## 2.2 Communication in Sensor Networks

Communication in a sensor network can be viewed as a tree, with the root being the base station. Synchronizing the transmission between nodes on a single path to the root is crucial for efficient in-network aggregation. A sensor (parent) needs to wait until it receives data from all nodes routing through it (children) before reporting its own reading. This delay is needed so that the parent node  $p$  can combine the partial aggregates reported by its children with its own reading and then send one message representing the partial aggregation of values sampled at the subtree rooted at  $p$ . The problem of deciding how long to wait (i.e., synchronize the sending and receiving of messages) is treated differently in Cougar and TAG.

Synchronization in TAG is accomplished by making a parent node wait for a certain time interval before reporting its own reading. This interval, called a *communication slot*, is based on subdivisions of the query period, which is referred to as an *epoch*. During a given communication slot, there will be one level of the tree sending and one level listening. In the following slot, those that were sending will go into *doze* or *sleep* mode until the next epoch, while the nodes that were receiving will now be transmitting. The cycle continues until all levels have sent their readings to their parents. When a parent receives the information, it aggregates the information of all children along with its own readings before sending the aggregate further up the tree. This synchronization scheme provides a query result every epoch duration.

Synchronization in Cougar is motivated by the fact that for a long running query, the communication pattern between two sensors is consistent over short

periods of time. Hence, in a certain round, if node  $p$  receives data from a node  $c$ , then it will realize it is the parent of that node  $c$ . Node  $p$  will add  $c$  to its *waiting list* and predict to hear from it in subsequent rounds. In the following rounds, node  $p$  will not report its reading until it hears from all the nodes on its waiting list. However, one case where this prediction fails is when the reading gathered by node  $c$  does not satisfy a certain selection predicate and hence needs to be discarded. In this case, under the Cougar protocol, node  $c$  will send a *notification packet* to prevent node  $p$  from waiting on  $c$  indefinitely.

### 2.3 Temporal Coherency-Aware In-Network Aggregation

TiNA (short for **T**emporal coherency-aware **i**n-**N**etwork **A**ggregation) is built as a layer that operates on top of in-network aggregation systems in order to minimize energy consumption throughout the entire sensor network [16]. The current implementation of TiNA has been designed to work with both TAG and Cougar.

TiNA selectively decides what information to forward up the routing tree by applying a hierarchy of filters along each path of the network. The selectivity of TiNA is based on a user specified *TOLERANCE* ( $tct$ ). The  $tct$  value acts as an output filter at the readings level, suppressing readings within the range specified by  $tct$ . For example, if the user specifies  $tct = 10\%$ , the sensor network will only report sensor readings that differ from the previously reported readings by more than 10%. Values for  $tct$  range from 0, which indicates to report readings if any change occurs, to any positive number. This  $tct$  is the maximum change that can occur to the overall quality of data in the system using TiNA.

A TiNA sensor node must keep additional information in order to utilize the temporal coherency tolerance. The information kept at a certain sensor depends on its position in the routing tree (i.e., a leaf or an internal node). Leaf nodes keep only the *last reported reading* which is defined as the last reading successfully sent by a sensor to its parent. Internal nodes, in addition to the last reported reading for that node, keep the last reported data it received from each child. This data can either be a simple reading reported by a leaf node or a partial result reported by an internal node.

Having the last operation repeated at every parent node along all the network paths provides a hierarchy of filters on every path. Setting the  $tct$  to zero for the hierarchical filtering at intermediate nodes ensures that partial aggregates, and eventually final aggregates, are always within the user-specified  $tct$ .

The hierarchy of filters TiNA provides is important for the incremental processing of aggregate queries as it captures cases of temporal correlation that cannot be captured at the readings level by individual sensors. For example, consider the aggregation function SUM; readings from different sensors might change from one round to another, however, it is possible that the overall sum stays the same. This can only be detected at a parent node which intercepts the stream of readings generated by these sensors and acts as an intermediate *centralized* stream processor. Note that this intermediate stream processing can provide a completely empty partial result or a partial result that is missing few aggregate groups when compared to the old partial result. In both cases, this node relies on the fact that its parent stored its last reported data and it will use it to supply the missing groups.

### 3 ENERGY EFFICIENT DATA ROUTING IN SENSOR NETWORKS

In this work, we assume a sensor grid environment in which the transmission range of each sensor node is one hop (i.e., all neighboring nodes are of equal distance and consume the same transmission energy). This is done to simplify the presentation and to streamline the evaluation of our proposed method. However, our proposed method is directly applicable to the general case (of non-uniform sensor network configurations) as well.

The ability to route data from the various nodes of the sensor network towards a central sink point (i.e., the base station) is fundamental to the operation of sensor networks. To support routing of data, the sensor network is configured into a routing tree, where each node (child) selects a *gradient* [2] or *parent* [10] to propagate its own readings.

The sensor network constructs the routing tree along with the propagation of the query. We assume that a new query in our model originates at the base station which forwards it to the nearest sensor node. This sensor node will then be in charge of disseminating the query down to all the sensor nodes in the network and to gather the results back from all the sensor nodes.

Traditional network configuration methods rely on link strength to construct the routing tree [18]. A child will pick the parent with the highest link strength, since this would usually correspond to shorter distance and thus less energy for transmitting data to the parent.

**First-Heard-From Network Configuration** The First-Heard-From (FHF) Network Configuration method is a simple way for children to choose parents and thus establish the routing tree. This method is derived from the link strength approach, when the sensor network follows a grid model and the transmission range of sensor nodes is one hop.

The basic idea behind the FHF network configuration algorithm is as follows. Starting from the root node, nodes transmit the new query. Children nodes will select as their parent the first node they hear from and continue the process by further propagating the new query to all neighboring nodes. The process terminates when all nodes have been “connected” via the routing tree.

The FHF method is formally described as follows:

1. The root sensor prepares a query message which includes the query specification. The root sensor also sets the ( $L_s$ ) value in the message to its level value (i.e.,  $L_{root}$  which is 0 initially). It then broadcasts this query message to the neighboring sensors.
2. Initially, all sensor nodes have level values set to  $\infty$ . A sensor  $i$  that receives a query message and has its level value currently equal to  $\infty$  will set its level to the level of the node it heard from, plus one. That is,  $L_i = L_s + 1$ .
3. Sensor  $i$  will also set its parent value  $P_i$  to  $Id_s$ . It then will set  $Id_s$  and  $L_s$  in the query message to its own  $Id_i$  and  $L_i$  respectively and broadcast the query message to its neighbors.
4. Steps 2 and 3 are repeated until every node  $i$  in the network receives a copy of the query message and is assigned a level  $L_i$  and a parent  $P_i$ .

This routing scheme is simple yet highly effective. It creates a path whereby child nodes can propagate readings up to the root. It also creates a way in which a query message from the root can be received by all nodes in the network. In addition, each node has been assigned a level which is needed for synchronization methods such as the epoch scheme in TAG.

The main weakness of this method is that it creates the network in a random way (only based on network proximity). The children assign parents based on whichever node happened to broadcast the routing message first. This method fails to consider the semantics of the query or the properties/attributes of the

sensor nodes and hence it cannot take any opportunities for energy savings. In the next section we present our proposal for an improved network configuration method that alleviates these problems and saves energy.

#### 4 GROUP-AWARE NETWORK CONFIGURATION

In order to have a network configuration method that considers the semantics of the query and the properties of the sensor nodes, we look closely at how in-network aggregation works. In-network aggregation will depend on the query *attributes* and the *aggregation function*. On the one hand, the list of attributes in the Group-By clause subdivides the query result into a set of groups. The number of these groups is equal to the number of combinations of distinct values for the list of attributes. Two readings from two different sensor nodes are only aggregated together if they belong to the same group. On the other hand, the aggregation function determines the structure of the partial aggregate and the partial aggregation process.

For example, consider the case where the aggregate function is SUM. In this case, the partial aggregate generated by a routing sensor node is simply the sum of all readings that are forwarded through this sensor node. However, if the aggregate function is AVERAGE, then each routing sensor node will generate a partial aggregate that consists of the sum of the readings and their count. Eventually, the root sensor node will use the sum and count to compute the average value for each group before forwarding it to the base station for further processing and dissemination.

Because aggregation combines all the readings for a particular group into one aggregate reading, creating a routing tree that keeps members of the same group within the same path in the routing tree should help decrease the energy used. The reason is simple: by “*clustering*” along the same path nodes that belong to the same group, the messages sent from these nodes will contain less groups (i.e., be shorter, thus reducing communication costs).

##### 4.1 Example of Group-Aware Network Configuration

To better illustrate the basic motivation, benefit, and reasoning behind group-aware network configuration, consider the example shown in Figure 1. In this figure, nodes 2, 4, and 6 (the shaded ones) belong to one group, whereas nodes 1, 3, 5, and 7 belong to a different group. Under the standard FHF network configuration (Figure 1a), nodes 4 and 5 could pick 2 as their parent, whereas

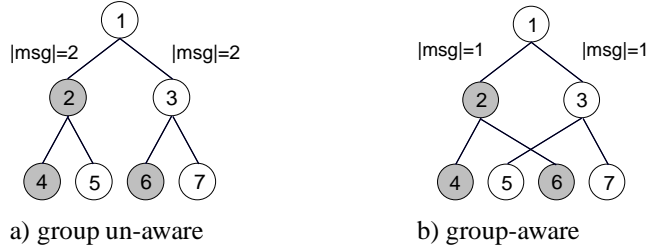


Figure 1: Benefits of group-aware network configuration

nodes 6 and 7 could pick 3 as their parent. Using in-network aggregation, the message sizes from nodes 2 and 3 to the root of the network will both be 2 tuples (i.e., contain partial aggregates from two groups). On the other hand, if we were able to cluster along the same path nodes that belong to the same group (Figure 1b) we would reduce the size of messages from nodes 2 and 3 in half: each message will only contain the partial aggregate from a single group (1 tuple). Next, we present the proposed algorithm, which achieves such clustering.

#### 4.2 GaNC Protocol

Our proposed *Group-Aware Network Configuration* method, or *GaNC*, constructs the routing tree as follows:

1. The root sensor prepares a query message which includes the query specification. The root sensor also sets the ( $L_s$ ) value in the message to its level value (i.e.,  $L_{root}$ , which is initially set to 0). It also sets the ( $G_s$ ) to be its group id. It then broadcasts this query message to the neighboring sensors.
2. A sensor  $i$  that receives a query message and has its level value currently equal to  $\infty$  will set its level to the level of the node it heard from, plus one. That is,  $L_i = L_s + 1$ .
3. Sensor  $i$  will also set its parent value  $P_i$  to  $Id_s$  and its parent's group id  $PG_i$  to  $G_s$ . It will then set  $Id_s$ ,  $L_s$  and  $G_s$  in the query message to its own  $Id_i$ ,  $L_i$  and  $G_i$  respectively and broadcast the query message to its neighbors.

4. While there are still query messages being propagated around the network, node  $i$  continues to listen to all messages it can hear.
5. If node  $i$  hears a message from a node at the same level as itself minus one ( $L_i - 1$ ), it uses *tie-breaker conditions* to decide if this new node should become its new parent. If so, node  $i$  makes  $Id_s$  its new parent.
6. Steps 2-5 are repeated until all query messages in the network have been sent out and received.

The GaNC algorithm is similar to the FHF algorithm. The main difference is that a child under the GaNC method can switch to a “better” parent while the tree is still being built. This switch is based on a set of *tie-breaker conditions* that go beyond the network characteristics and introduce the semantics of aggregation.

The goal of the the GaNC algorithm is to incorporate group identity into the routing tree construction. As such, the first tie-breaker condition (for Step 5 of the algorithm) is whether the child has the same group id as the parent. As long as a child is within listening distance of multiple parent choices, a child will choose a parent that has the same group id as itself instead of a parent from a different group. This is a choice that will allow parents and children to be in the same group as much as possible.

In the general case, a sensor node will be within listening range of multiple other nodes. Despite the savings in clustering nodes of the same group along the same path, a node that is far away will require significantly more transmission energy, and as such is not a good candidate. For that reason, we introduce a *distance factor*,  $df$ , that will limit the maximum range for which we consider candidate sensor nodes (for coming up with a “better” parent node). Under this approach, if  $d_i$  is the shortest distance seen so far (based on an estimation from signal strength), we will only consider nodes whose distance from a child node is at most  $df \times d_i$ , for example, for  $df = 1.2$  we will only allow up to 20% more than the minimum distance.

Thus, the second tie-breaker is the estimated distance (or link quality) from the child to the parent. The parent with the lowest distance will be chosen in cases when there is more than one parent to choose from (that is in the same group as the child), or when no parents are in the same group as the child. The

reason for this is that in both cases, routing through the closest parent will save transmission energy for the child.

#### 4.3 GaNCi: GaNC Improvement

As an improvement on the original GaNC algorithm we also looked at allowing the child to choose a parent from a larger selection of nodes. In the original algorithm (Step 5 above) a child would consider a node as a possible parent if its level was that of the child minus one. The idea behind this constraint was that a child should always try to get one level closer to the root when choosing a parent. However, this limits the number of choices for selecting a parent and hence reduces the chances of the node finding a parent in the same group as itself.

In GaNCi (GaNC improved), the improvement we made was to change Step 5 to consider nodes that are in the same level as itself in addition to those in level lower than itself. In essence, the child can now choose a parent both from potential parents as designated in the original algorithm and from its own siblings. The benefits from this should be that more nodes have a better chance of having a parent in the same groups as themselves. This improvement should even surpass that of original GaNC, solely because there is a greater chance of children being in the same group as their parent.

In order to prevent siblings from selecting each other as their parent, which will prevent any information from those nodes or nodes routed through them from being propagated to the root, GaNCi requires each parent node to maintain a *child list*, much like the waiting list in Cougar. When a child chooses a new parent, it broadcasts a message letting all nearby sensors know of the change. Using the information from this message, both the previous parent (if one existed) and the newly identified parent for the child can update their child lists appropriately. This thereby accomplishes both tasks required by the new protocol: (1) allowing parents to create an up to date list of children it has and (2) allowing old parents to remove children that have switched to a new parent. To incorporate these changes into GaNCi, Step 5 of GaNC protocol is replaced with the following:

- 5 a. If node  $i$  hears a message from a node  $n$  at the same level as itself or lower ( $L_n \leq L_i$ ) and  $n$  is a child of  $i$  and is announcing a new parent, remove  $n$  from child list of  $i$ .
- b. If node  $i$  hears a message from a node  $n$  at the same level as itself or

lower ( $L_n \leq L_i$ ) and  $n$  is not a child of  $i$ , use *tie-breaker conditions* to decide if this new node should become the new parent and notify parent  $n$  that  $i$  has chosen it.

It is important to note that this improvement can be immediately incorporated into a scheme such as Cougar which is already keeping a waiting list of its children. In a scheme like TAG, the sensor node will have to be modified so that it can keep track of all known children.

## 5 EVALUATION TESTBED

In order to empirically study in-network aggregation in sensor networks, we created a simulation environment using CSIM [15]. Following typical sensor network simulation practices, the simulated network was configured as a grid of sensors. Each node could transmit data to sensors that were at most one hop away from it. In a grid this means it could only transmit to at most 8 other nodes. We simulated a contention-based MAC protocol (PAMAS) which avoids collision [17]. In this protocol, a sender node will perform a carrier sensing before initiating a transmission. If a node fails to get the medium, it goes to sleep and wakes up when the channel is free.

In the following experiments, we used the Group-By query format as described in Section 4, with the network producing results at intervals defined in the query. The size of the sensor networks is varied between 15x15 and 45x45, and the type of aggregation being performed is SUM. The group ids of the sensor nodes are based on randomness and the number of groups is varied between 2 and 50. These experiment parameters along with all other parameters are summarized in Table 1.

### 5.1 Random Walk

We used the random walk model for data generation. In this model, the domain of values was between 1 and 100 (to approximate temperature readings in Fahrenheit). A sensor reading is generated once at the beginning of each query interval. The value changes between one interval to the next with a probability known as the *randomness degree* ( $RD$ ). Each time a sample is to be generated, a coin is tossed. If the coin value is less than  $RD$ , then a new value is generated, otherwise the sample value will be the same as before. For example, if  $RD = 0.0$ , then the value sampled by a sensor will never change, while if  $RD = 0.5$ , then there is a 50% chance that the new value at time  $t$  is different

Parameter	Value	Default
Grid Size	15x15 – 45x45	15x15
Number of Groups	2 – 50	15
Aggregate	SUM	
TiNA <i>tct</i> amount	0% – 30%	
Number of Epochs	100	
Routing Scheme	FHF, GaNC	FHF
Randomness Degree	0.0 – 1.0	0.5
Random Step Size Limit	10%–50% of domain	10%

Table 1: Simulation Parameters

from the value at time  $t+1$ . We used the *Random Step Size Limit* to restrict how much the new value can deviate from the previous value. This limit is expressed as a percentage over the domain of values. In our case, a 10% limit implies that a new reading can differ by at most 10 (=10% of 100) compared to the previous reading.

## 5.2 Performance Metrics

In our experiments we focused on two measurements: energy consumption and relative error.

**Energy:** Energy is consumed in four main activities in sensor networks: transmitting, listening, processing, and sampling. We focused on transmission and listening power, since the amount of time spent sampling is the same for all techniques. We did not include energy required for processing because it is negligible compared to that needed for communication.

As mentioned before, a sensor node will send its data to the root through its assigned parent. A parent node is one hop away from its child, and one hop closer to the root than its child. So every node sends its data exactly one hop away, all of which are the same distance from one another. This allows us to assume a uniform cost of transmitting data. However, the overall energy consumed to transmit a partial result depends on the size of the partial results and the number of messages.

The values of the parameters needed to calculate the transmission cost were the same as in [4]. Specifically, we simulated sensors operating at 3 Volts and capable of transmitting data at a rate of 40 Kbps. The transmit current draw is

0.012 Amp while the receive current is 0.0018. Hence, the cost of transmitting one bit in terms of energy consumption units (*Joules*) is computed as:

$$T_{cost} = 3 \text{ Volt} * 0.012 \text{ Amp} * 1/40,000 \text{ Sec} = 0.9 \mu\text{Joules}.$$

The cost of listening for one second is computed as:

$$R_{cost} = 3 \text{ Volt} * 0.0018 \text{ Amp} = 0.0054 \text{ Joules}.$$

The energy consumed during listening is independent of the number of messages received by the sensor. It only depends on the time spent by the sensor being active and listening. Cougar does not specify when a sensor stops listening and switches to doze mode. Hence, in our simulation, we assumed that each sensor will start listening at the beginning of each round. After a sensor receives data from all nodes on its waiting list it will switch to doze mode.

**Relative Error Metric:** The *relative error metric* (REM) is a measure of how close the exact answer and the approximate answer are. The exact answer is generated if all sensors deliver their current readings within the each epoch interval. An approximate answer is one where some sensors decide not to send their reading. In our experiments, a sensor decides not to send a message because of the user-specified temporal coherency tolerance when TiNA is used.

In order to compute REM, we first need to measure the error over the Group-By query. We measured this error as described in [14]. Assume a query aggregates over a measure attribute  $M$ . Let  $\{g_1, \dots, g_n\}$  be the set of all groups in the exact answer to the query. Finally, let  $m_i$  and  $m'_i$  be the exact and approximate aggregate values over  $M$  in the group  $g_i$ . Then, the error  $\epsilon_i$  in group  $g_i$  is defined to be the relative error, i.e.,  $\epsilon_i = \frac{|m_i - m'_i|}{m_i}$ . The error  $\delta$  over the Group-By query is defined as:  $\delta = \frac{1}{n} \sum_{i=1}^n \epsilon_i$  finally, the average REM (or simply, REM) over time is defined as:

$$REM = \frac{1}{T} \sum_{t=1}^T \delta_t$$

where  $\delta_t$  is the query error at  $epoch_t$ .

We are using REM as indication of the *quality of data* (QoD), where a high REM reflects a low QoD, while a low REM corresponds to a high QoD. Hence, we will be using both terms interchangeably.

## 6 EXPERIMENTS AND RESULTS

Using the simulator, we performed extensive experiments to evaluate the performance of our two algorithms GaNC and GaNCi. We compared GaNC

and GaNCi with the FHF network configuration scheme presented in Section 3. Given that Cougar and TiNA require no modifications to use our algorithms, in this section we present our experiments with these two schemes.

### 6.1 Effects of Group-Aware Network Configuration

In this experiment we compare FHF with GaNC and GaNCi for varying  $tct$  amounts. We will show the effects of GaNC for network sizes of 15x15 and 45x45 and report both the energy savings and the quality of data.

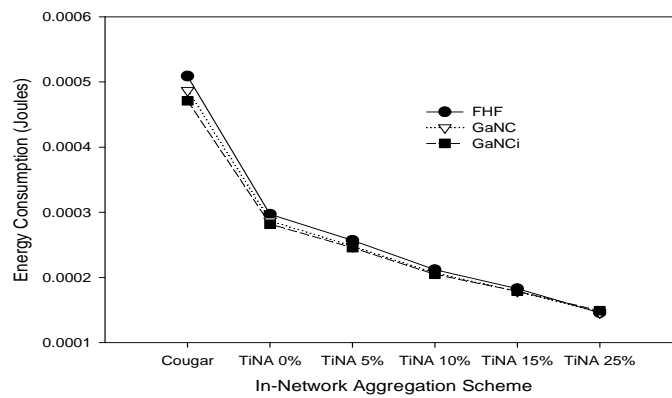


Figure 2: GaNC in 15x15 Grid (Energy)

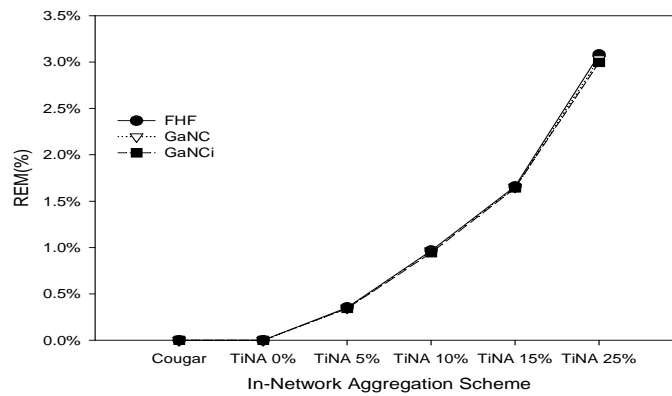


Figure 3: GaNC in 15x15 Grid (REM)

Figures 2 and 3 show the energy and relative error for the 15x15 size network and Figures 4 and 5 show the same for the 45x45 size network. The number of groups used in this experiment was set at 15.

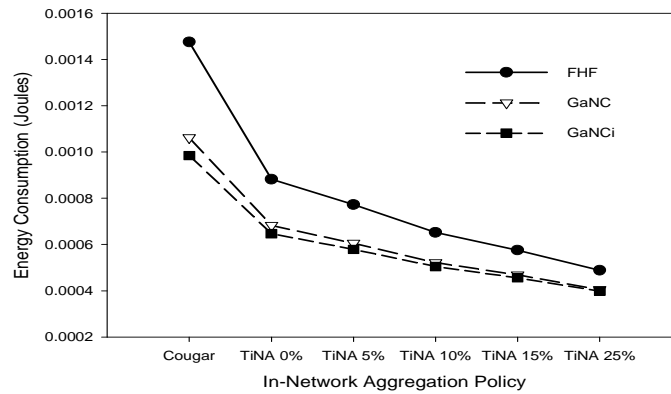


Figure 4: GaNC in 45x45 Grid (Energy)

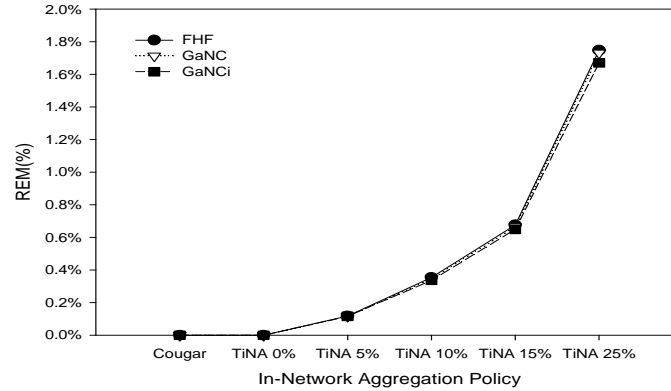


Figure 5: GaNC in 45x45 Grid (REM)

The first observation is that for the most part, using GaNC decreases the amount of energy used by the sensor network. This is especially prominent in larger networks. For TiNA with  $tct=0%$ , energy savings are 3.4% for the 15x15 grid and 23.6% for the 45x45 grid. The savings when using GaNC with

plain Cougar are even higher: 4.4% for the 15x15 grid and 28.1% for the 45x45 grid. This shows that the proposed group-aware network configuration method can reduce energy significantly when used with Cougar and can even help save additional energy when used in conjunction with the TiNA framework.

A related observation that can be made is about the relationship between GaNC and GaNCi for energy savings. By giving the child more choices for selecting a parent, there is an even better chance the child will fall into the same group as its parent. This should bring even greater energy savings, which the experiments show is true. For the  $tct=0\%$  and grid size of 15x15, GaNCi saves 5.1%, an additional 1.7% over normal GaNC. For Cougar and the 15x15 grid, GaNCi saves 7.5%, an additional 3.1%. These additional savings are even greater for the 45x45 sized grid. For  $tct=0\%$  and Cougar, GaNCi saves 26.7% and 32.3% respectively over FHF. These are additional savings of 3.1% and 4.2% over normal GaNC, showing that the GaNC improvement does help increase the amount of energy saved.

The next observation is that regardless of the size of the network, the energy savings of GaNC over FHF decrease as the  $tct$  of TiNA increases. In fact, for the 15x15 network, there is cross-over point where FHF requires the same amount of energy as GaNC and both require less energy than GaNCi. This is illustrated in Figure 2 where GaNC was saving 3.4% and GaNCi was saving 5.1% over FHF for  $tct=0\%$ . When  $tct$  is increased to 25%, GaNC is using the same amount of energy as FHF and GaNCi is using 2% more energy than either FHF or GaNC. This result is not entirely surprising, however. When using GaNC some nodes will switch to parents that are in the same group as themselves. While this tends to decrease the number of message a parent sends, there are cases where switching parents can cause more messages to be sent. This is increased with GaNCi which has an even better chance of changing parents.

For example, assume that, using FHF, two children of the same group are routed through a parent of a different group. This would result in 3 messages being sent overall (one from each child and one from the parent further up the tree). If, however, when GaNC is used, each of these children change to be with a parent of their same group, they may end up choosing two different parents, because the parents in their same group are not close enough to be “clustered” together. In order to propagate information up the tree under this setup, 4 messages are needed (one from each child and one from each parent.

This is a 25% increase in the total number of messages, which in turn causes an increase on the energy used in the network.

For larger networks, the positive effects of using GaNC will outweigh the negative effects (per our previous example). As the  $tct$  increases, there are less nodes that are transmitting, since their value changes are not violating the specified  $tct$ . In larger networks, since there are many nodes, there will still be a lot of nodes transmitting, even under high  $tcts$ . In Figure 4, we can see that for the 45x45 grid, the energy savings at  $tct=0\%$  are 23.6% and they drop to down to 17.1% for  $tct=25\%$  (however, there is no cross-over point in this case).

The final observation is that there is very little difference in the relative error between using GaNC versus using FHF to create the routing tree, even with the large savings in energy (from GaNC). For the 15x15 grid, the relative error has decreased in the case of GaNC. This decrease is minor, ranging from .005% for  $tct=5\%$  to .04% for  $tct=25\%$ , but exists nonetheless. These decreases are even larger for GaNCi, where relative error has decreased between .01% for  $tct=5\%$  to .08% for  $tct=25\%$ . This improvement is due to the "free" ride some parent nodes may get by having a child already sending the same group as the parent and therefore be able to aggregate its own reading into the group aggregate without adding to the amount of energy used.

In the case of the 45x45 grid, the relative error again decreases by a small amount. This decrease was between .001% for  $tct=5\%$  and .016% for  $tct=25\%$ . Again, for GaNCi, the decrease is a little larger with decreases ranging from .002% for  $tct=0\%$  to .077% for  $tct=25\%$ . We have observed, however, that with smaller number of groups the relative error can also increase by very small amounts. This small increase is explained by multiple nodes counteracting each others' changes; thus the internal node does not get the "free" ride it may have gotten with only one child changing values.

## 6.2 Synergy of TiNA and GaNC under varying number of groups

In this experiment we examine how the number of groups affects the behavior of GaNC. Figure 6 shows the results from this experiment. We compared plain Cougar, TiNA over Cougar with  $tct=0\%$ , and TiNA over Cougar with  $tct=10\%$ . We ran two sets of experiments, one where FHF was used for configuring the network and one where we used GaNC instead (we note such cases with +GaNC). We do not show the runs with GaNCi in this experiment because the general pattern is the same as for GaNC, the only difference is the

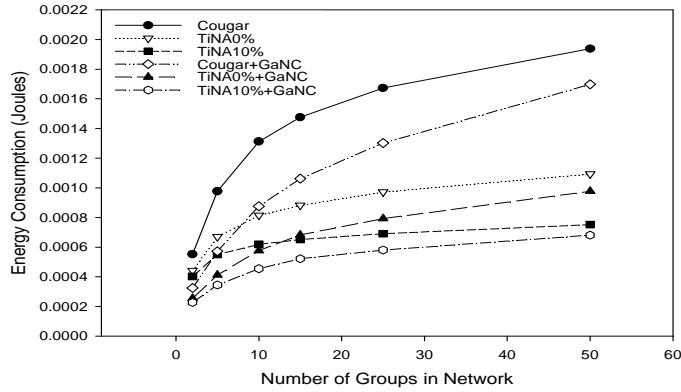


Figure 6: Energy Comparison for varying number of groups

line would be slightly lower. We did this to make the figure easier to read.

The number of groups in the experiment ranged from 2 to 50. We had a total of  $45 \times 45 = 2025$  nodes. With just 5 groups, GaNC is expected to reduce energy consumption significantly, since children nodes will be able to select parents that are in the same group as them (in other words, GaNC will have a lot of options to choose from). For all three cases, using GaNC instead of FHF saves 41.4% in energy for Cougar, 38.5% for TiNA(0%), and 37.4% for TiNA(10%).

Using GaNC when the number of groups is 50 will not reduce energy as dramatically as with the case of 5 groups. When the number of groups is high, there are less chances that a child can find a parent in the same group. Based on the grid configuration (which we used in our simulations), each child has a maximum of three different parents to choose from. With 50 different groups, the chance that the child is in the same group as one of those three parents is less than 1%, so the savings will be minimal.

This can be observed from Figure 6: for 50 groups the savings with GaNC are only 12.4% for Cougar, 10.7% for TiNA(0%), and 9.5% for TiNA(10%). Overall, we see that when the number of groups increases, the savings with GaNC also decrease, but are still significant even when there are 50 different groups and the chance of a node actually switching parents in an efficient way is less than 1%.

## 7 RELATED WORK

The idea of exploiting the application semantics for data routing in sensor networks has been presented in [20, 9], where the goal is to use information-directed routing in order to minimize communication cost while maximizing information aggregation. The work in [9] showed the significant gains of applying information-directed routing in locating and tracking moving objects.

Using the query semantics for efficient data routing was introduced in [11], which proposed the use of a semantic routing tree (SRT). The basic idea that motivates the use of SRT is the fact that a given query does not apply to all nodes in the network. Hence, those nodes for which the query does not apply can be excluded from the query in order to save communication costs. As in GaNC, the work on SRT considered optimizing the routing tree for the special case of constant-valued attributes (e.g., location). However, the objective of the SRT is providing a design to minimize the number of nodes participating in a query with a predicate over that constant-valued attribute. Instead, in GaNC, the objective is to cluster along the same path sensor nodes that belong to the same group in order to maximize the benefit of in-network aggregation in reducing the size of messages, even if all nodes participate in the query.

## 8 CONCLUSIONS

In this work we explored the idea of *influencing the construction of the routing trees for sensor networks* with the goal of reducing the size of transmitted data in networks with in-network aggregation, hence providing additional energy efficiency in sensor networks. We proposed two network configuration algorithms for sensor networks, called *GaNC* and *GaNCi*, that achieve energy savings by considering the semantics of Group-By queries and the properties of the sensor nodes. These two algorithms work synergistically with existing in-network aggregation methods but differ with respect to their applicability. GaNC can be immediately combined with existing schemes such as Cougar and TAG. On the other hand, GaNCi which can be readily combined with schemes such as Cougar, may require additional features to work with schemes such as TAG.

We have shown experimentally that our algorithms result in large savings in energy over typical in-network aggregation methods. In addition, we looked at using GaNC and GaNCi in conjunction with TiNA which is another scheme

for saving energy in-network aggregations methods. Our results have shown that GaNC can save up to 33% in energy over existing in-network aggregation schemes and can save an additional 29% over the savings of TiNA, when used in tandem with it. We also showed that using GaNC and GaNCi with TiNA can help improve the quality of data in systems with imperfect quality of data.

#### REFERENCES

- [1] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proc. of MDM*, 2001.
- [2] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proc. of SOSIP*, Oct 2001.
- [3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, Jan 2000.
- [4] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro.*, 22(6), 2002.
- [5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc of ASPLOS*, 2000.
- [6] C. Intanagonwiwat et al. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. of MOBICOM*, Aug 2000.
- [7] P. Juang et al. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. In *Proc. of ASPLOS'02*.
- [8] C. Lin, C. Federspiel, and D. Auslander. Multi-sensor single actuator control of hvac, 2002.
- [9] J. Liu, F. Zhao, and D. Petrovic. Information-directed routing in ad hoc sensor networks. In *2nd ACM international conference on Wireless sensor networks and applications*, 2003.
- [10] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proc. of OSDI*, 2002.
- [11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. of ACM SIGMOD*, 2003.
- [12] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of ACM WSNA'02*, 2002.
- [13] C. Perkins. Ad-hoc on demand distance vector routing (AODV). Internet-draft, Nov. 1997.

- [14] S. Acharya, P. B. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *Proc of ACM SIGMOD*, 2000.
- [15] H. Schwetman. CSIM user's guide. MCC Corp.
- [16] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Tina: A scheme for temporal coherency-aware in-network aggregation. In *Proc. of MobiDE*, 2003.
- [17] S. Singh and C. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Comm. Review*, 28(3).
- [18] A. Woo and D. Culler. A transmission control scheme for media access in sensor networks. In *ACM Mobicom*, July 2001.
- [19] Y. Yao and J. Gehrke. Query processing for sensor net. In *Proc. of CIDR*, 2003.
- [20] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, vol. 19, 2002.