Building Trees Based On Aggregation Efficiency in Sensor Networks

Albert F. Harris III, Robin Kravets, and Indranil Gupta
University of Illinois at Urbana-Champaign
email:{aharris,rhk,indy}@cs.uiuc.edu

***Abstract*— Sensor network protocols must minimize energy due to their resource-constrained nature. Large amounts of redundant data are produced by the sensors in such networks, however sending unnecessary data wastes energy. One common technique used to reduce the amount of data in sensor is data aggregation. Therefore, we consider the impact and cost of data aggregation in sensor networks to achieve energy efficient operation. We propose a new notion of *energy efficiency* that can be used to decide where aggregation points in the network should be placed. The main factor affecting energy efficiency is the location of the data aggregation points. The optimal choice of these points is determined by the aggregation efficiency, which determines the amount of data reduction. We present our aggregation tree algorithm "Oceanus" that produces energy-efficient aggregation trees by taking into account the aggregation efficiency. Our evaluation shows that by using aggregation efficiency, Oceanus provides higher energy efficiency compared to existing solutions for data aggregation.**

I. Introduction

Advances in computing and communication technologies have enabled the creation of small devices capable of complex sensing and computation. While the goal is to embed these devices into our surrounding environments, energy consumption has become the main limiting factor of the lifetimes, and so effectiveness, of these sensor networks. To support increased network lifetime, it is necessary to design energy-efficient communication protocols. Although such protocols have been proposed in the context of ad hoc networks [17], the data-centric focus of sensor networks lends itself to better energy efficiency through intelligent management of the data.

In typical communications scenarios for sensor networks, data about a particular event is collected by the sensors and is then sent to a data sink, which can be anywhere in the network. Frequently, the sink may not require the original data from each individual sensor, but instead only require an aggregate function (*e.g.*, sum, average, etc.) of the collected data from all sensors. The benefit of such *data aggregation* is that it can reduce the total amount of data sent through the network, increasing network performance and decreasing energy consumption. However, the overall effectiveness of data aggregation is dependent on where and when the aggregation actually occurs. Although several data aggregation algorithms and frameworks have been proposed [1], [10], [12], [13], [19], finding the optimal aggregation points in the network is still an open area of research.

Data aggregation changes the communication in the network by allowing individual nodes to collect data samples from multiple sources and combine them to be transmitted as one sample. Energy can be saved if the overall amount of data transmitted in the network is reduced by the aggregation. Therefore, the *energy efficiency* of such aggregation is affected by two metrics *aggregation efficiency* and *aggregation cost*. Aggregation efficiency captures the amount of data compression achieved by the aggregation function. If the aggregation of n data samples results in one new data sample, the aggregation efficiency is said to be perfect. However, if the result is simply the n samples concatenated together, the aggregation efficiency is poor, and only benefits from merging headers. Although aggregation may be highly efficient and so significantly reduce the amount of data transmitted, it is also necessary to consider the computational cost of the aggregation in the node. While some aggregation may be cheap (*e.g.*, simple sum), some aggregation may be computational expensive (*e.g.*, combining audio samples).

Since the goal of data aggregation is to reduce redundancy in the communication, the best-suited delivery network is a tree, where aggregation occurs when two branches merge. The challenge, therefore, is to design algorithms that understand aggregation efficiency and cost to create trees with the most energy-efficient aggregation points. As discussed in Section III-B, if the aggregation algorithm is perfect (*i.e.*, perfect efficiency and 0 cost), the optimal aggregation tree is a Steiner Tree. Given an imperfect aggregation algorithm (*i.e.*, less efficient and some cost), the optimal aggregation tree is a Weighted Steiner Tree. Although calculating a Steiner Tree, weighted or unweighted, is NP-complete [4], it is possible to use some heuristics to approximate the target Steiner Tree and use this approximation as the aggregation tree in the sensor network.

Current heuristic-based aggregation tree algorithms use either opportunistic methods (*e.g.*, Directed Diffusion [8]) or greedy incremental methods (*e.g.* Intanagonwiwat, *et. al.* [7], [9]). In opportunistic methods, data flows through shortest paths from the sources to the sink. In the event that paths meet, the paths are joined to form an aggregation point. Such aggregation points tend to be close to the sink because shortest path flows from different sources to the same sink intersect downstream. In the greedy incremental methods, one source initiates a shortest path flow to the sink. Then, the other sources connect to that path via shortest paths, which generally results in aggregation points closer to the sources. However, the efficiency of the greedy incremental method is entirely determined by the first path and can result in very inefficient aggregation trees. The main problem with both of these methods is that they cannot consider aggregation cost, and so only approximate a Steiner Tree.

To find the most energy-efficient aggregation tree, it is necessary to understand the energy efficiency of the data aggregation algorithm. In this paper, we use our formulation of aggregation efficiency and cost to explore the energy efficiency of data aggregation. Essentially, we show that if the aggregation efficiency is perfect and the cost is free, the aggregation points should be as close as possible to the sources to save the most energy. However, as the aggregation efficiency degrades or the cost increases, the optimal aggregation points drift towards the sink, since the savings from the reduced communication no longer outweighs the extra cost of aggregation.

The main contribution of our research is the design and analysis of Oceanus, a heuristic-based aggregation tree algorithm that approximates the optimal Weighted Steiner Tree for a given aggregation efficiency and cost. By understanding the tradeoffs between aggregation efficiency and cost, Oceanus creates trees with aggregation points closer to the sources when efficiency is high and cost is low and trees with aggregation points closer to the sink when efficiency is low and/or cost is high. Our evaluation of Oceanus shows that for most aggregation scenarios, Oceanus saves energy over the shortest path tree, the opportunistic method and the greedy incremental method. We also notice that in extreme cases, where the sources are topological isolated from each other, the opportunistic method outperforms all others, since little or no aggregation is possible. This is a limitation of the heuristic-based algorithm. However, in such cases, if the disjoint sources are treated independently, Oceanus can again outperform the other approaches.

The remainder of this paper is structured as follows. Section II discusses energy consumption in sensor networks in terms of aggregation efficiency and cost and presents a single metric that captures these relationships. Section III presents four aggregation tree algorithms: the optimal tree algorithm and the three heuristics algorithms, including Oceanus. Section IV presents the methodology used to analyze the aggregation efficiency space as well as the simulation setup for experimentation. Section IV-B presents our experimental results. Finally, Section V presents conclusions and future directions for our research.

II. Energy Efficiency in Sensor Networks

Energy efficiency is a driving concern in the design and implementation of sensor networks. When using data aggregation, there are three components to energy consumption in sensor networks: the energy consumed by control messages to set up the aggregation tree for a given event, the energy consumed by all data transmissions for a given event and the energy consumed by the aggregation of the data at the aggregation points. While the energy consumed by the control messages is relatively fixed for a given network, there is a direct tradeoff between the energy consumed by the data transmissions and the energy consumed by the aggregation. In this section, we define the energy efficiency of data aggregation, which captures this tradeoff. In the following section, we show how energy efficiency can be used to design aggregation tree algorithms.

For a given event, data must flow from the source to the sink, as determined by an initial interest notification sent from the sink. Many algorithms have been proposed to enable efficient interest notification ([1], [2], [5], [12], [13], [16], [19]). For the non-opportunistic approaches, some energy is consumed during coordination between the sources and the sink to set up the aggregation tree. Since sensor networks are relatively static, this paper focuses on the energy efficiency of static aggregation tree algorithms. If enough nodes die so that the current tree can no longer deliver the data, it is necessary to reconfigure the tree, incurring some energy consumption from control messages. We are currently investigating the impact of this control overhead and integrating it into a dynamic aggregation tree algorithm.

Once sinks and sources have been coupled in a sensor network, data can flow from the sources to the sink(s). The energy consumed at each hop of a flow is determined by the transmission rate R, the size of the header, H, and the size of the data, D. Therefore, the per-hop transmission energy of a flow can be defined as:

$$E_t = \frac{H + D}{R} \times P_{trans},$$

where P_{trans} is the device transmission energy. Whenever flows from the same event intersect, it is possible to combine their data into one flow. The resulting flow carries the aggregated data and only uses one header. For example, if n flows are being aggregated, where a packet from flow i is $[H_i, D_i]$, the packets from the aggregated flow are $[H_{aggr}, D_{aggr}]$, where $D_{aggr} = f(D_1, D_2, ..., D_n)$.

Although the goal is to reduce the number of bytes sent and so reduce the energy consumed by the transmission, it is possible that E_t could increase after the aggregation if $(H_{aggr} + D_{aggr}) > \sum(H_i + D_i)$. In other words, if the aggregation function results in an aggregate that is larger than the sum of the initial data sizes plus the size of $n - 1$ headers, the aggregation function should not be used. Therefore, we define aggregation efficiency as:

$$\xi_{aggr} = \frac{\sum(H_i + D_i)}{H_{aggr} + D_{aggr}}.$$

If $\xi_{aggr} > 1$, the aggregation reduces the number of bytes transmitted, and so reduces E_t. If $\xi_{aggr} < 1$, aggregation should not be performed.

While E_t captures the energy to transmit, it is also necessary to consider the cost of receiving the data and the cost of running the aggregation function. The cost of receiving the data is also a function of the data size D and is giving by:

$$E_r = \frac{H + D}{R} \times P_{recv}.$$

Therefore, reducing the data size via data aggregation will also reduce the cost of receiving the data by the downstream nodes. The cost of running the aggregation function (E_{aggr}) must be low enough so that it does not outweigh the savings in E_t from the aggregation. More formally, we define the energy efficiency of aggregation as follows:

$$\xi = \frac{\frac{\sum(H_i+D_i)}{R} \times (P_{trans} + P_{recv})}{\frac{(H_{aggr}+D_{aggr})}{R} \times (P_{trans} + P_{recv}) + E_{aggr}}.$$

If ξ is greater than one, data aggregation saves energy. Otherwise, it is cheaper to simply send the independent packets.

Energy efficiency obviously impacts the choice of an aggregation tree. Essentially, as the aggregation efficiency increases and the cost of the aggregation decreases, it is more efficient to aggregate close to the sources. As the aggregation efficiency decreases and the aggregation cost increases, the most efficient aggregation points migrate towards the sink. Current research, however, tends to look at algorithms for creating aggregation trees and aggregation energy efficiency in isolation. In the next section, we present various aggregation tree algorithms and discuss if and how they can integrate energy efficiency.

III. Aggregation Tree Algorithms

The focus of this work in on the effects of data aggregation efficiency on choosing the aggregation tree in sensor networks. There has been significant work in the areas of finding clusters of nodes to shift aggregation points among nodes to increase network lifetime [3], [6], [11], [15] and building general routing policy frameworks [10], but these are orthogonal to this work. In this section, we describe the optimal aggregation tree algorithm and then consider three heuristics used to construct aggregation trees in sensor networks. The three heuristics are opportunistic, greedy incremental, and Oceanus. In the following sections, we evaluate the effect of energy efficiency on each of these algorithms.

A. Link Energy Function

To find minimum cost aggregation trees, we model the energy cost of a link as a function of the energy required to aggregate at the sender (if aggregation is not performed, this is zero), the transmission energy expended by the sender, and the receive energy expended by the receiver, where D is the size of the data sent, including the packet header, R is the data rate, P_{trans} is the transmission power, P_{recv} is the receive power, and E_{aggr} is the aggregation energy.

$$E_{link} = E_{aggr} + \frac{D}{R} \times P_{trans} + \frac{D}{R} \times P_{recv} \quad (1)$$

This function captures the amount of energy for each data segment transmitted on a link. The sum of each link's energy consumption for transmitting data yields the total energy consumed by the aggregation tree.

B. Optimal Aggregation

The problem of finding a lowest cost aggregation tree can trivially be reduced to the problem of finding a Steiner tree in the graph. Formally, the Steiner tree of some subset of the vertices of a graph G is a minimum-weight connected subgraph of G that includes all of the vertices.

Let $G(V, E)$ be the set of all nodes in the sensor network with non-negative weights for each $e \in E$ corresponding to the cost to transmit data over link e. Let

$Z \subseteq V$ be the set of source nodes ($s_i \in S$, where $S \subset Z$) and the sink (k). The cost function is in terms of energy consumed on each link for transmission of data across that link. If the cost function for assigning the weights includes data aggregation (see Equation 1), finding the subnetwork $T \subseteq G$ such that every pair of vertices in Z is connected and the total cost of T is a minimum is equivalent to finding the minimum cost aggregation tree. However, this is the Weighted Steiner Tree.

The determination of a Weighted Steiner Tree is NP-complete [4]. Even if the edge weights are all equal (corresponding to a perfect aggregation algorithm), the problem remains NP-complete. Therefore, it is infeasible to calculate the optimal aggregation tree and heuristics for efficiently constructing aggregation trees are needed.

C. Opportunistic Aggregation

Opportunistic aggregation only aggregates streams if they happen to intersect on their way to the source, (*e.g.*, Directed Diffusion [8]). To achieve opportunistic aggregation, each source begins sending streams to the receiver via shortest path routes. As streams intersect, they are aggregated.

Aggregation points are always downstream, towards the sink. The more dispersed the source nodes are from each other in the network, the less likely aggregation is performed. Opportunistic algorithms tend to result in aggregation trees with aggregation points close to the sink. Such trees are beneficial if the energy efficiency of the aggregation is low, meaning that the savings from aggregating early does not outweigh the additional communication cost. However, opportunistic aggregation is less likely to result in energy efficient aggregation trees for aggregation functions with high efficiency and low cost.

D. Greedy Incremental Aggregation

The Greedy Incremental Aggregation algorithm (*e.g.*, [7]) begins by sending a single stream via a shortest path route to the sink. Each additional stream is then routed to a node participating in the first flow via a shortest path. This method prevents two streams from spanning the entire network only one hop apart.

Aggregation trees derived from this sort of algorithm often have aggregation points that lie somewhere in the middle of the network. This can yield significant efficiency gains over opportunistically created aggregation trees in cases where the aggregation algorithm has moderate to high energy efficiency.

```
CALCULATE-TREE( )
 G : Set of Nodes
  Z ⊆ G: Source Nodes + Sink Node
  S ⊆ Z: Source Nodes
  T: Nodes in Tree
  T = {}
  T′ ⊆ Z: Source and Sink nodes in Tree
  T′ = {}
  z ∈ S:z is chosen randomly
  x ∈ Z: x is closest to z
  Connect(x,z)
  T = T + x + z + path(x,z)
  T′ = T′ + z + x
 while T′ ≠ Z
 do
      x ∈ Z: z is closest to z ∈ T
      Connect(x,z)
      T = T + x + path(x,z)
      T′ = T′ + x
```

Fig. 1. Oceanus Aggregation Tree Algorithm

E. Oceanus

Oceanus approximates the aggregation tree providing the most energy-efficient communication using knowledge of the energy efficiency of the aggregation algorithm. Oceanus uses a heuristic-based algorithm that approximates a Weighted Steiner Tree, where the weights reflect the energy efficiency of the aggregation algorithm. To start, Oceanus randomly chooses one of the source nodes. It then finds the node in Z that is closest to the chosen node using a shortest path algorithm where the weights on the paths are given by Equation 1. These nodes are connected by this path. Then, the next node in Z that is closest to the tree that has been already formed is chosen and connected, and so on until a complete tree is obtained. The aggregation tree is calculated according to the algorithm depicted in Figure 1.

It is simple to see that this method results in each closest sensor node being connected via a least cost path, where cost is defined in terms of Equation 1. Since this is only a heuristic, it is possible that this is not the Weighted Steiner tree. If two nodes can be connected via two different least cost paths, the intermediate node that is chosen may be farther away from all of the remaining unconnected nodes than the other choice of intermediate node.

To find the aggregation tree, we assume that the sink node has knowledge of the sensor network topology. When an event of interest happens, each sensor node sends an initial notification of event to the sink node. At that time, the sink calculates the aggregation tree and informs the source nodes and the aggregation nodes of the

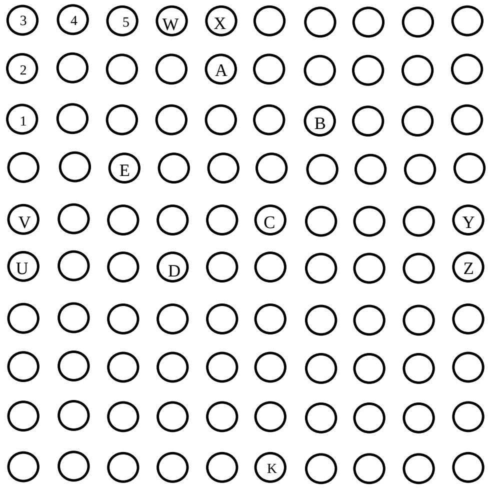

Fig. 2. Network Event Scenarios (Scenario 1: Nodes 1-5 are sources, Scenario 2: Nodes A-E are sources, Scenario 3: Nodes U-Z are sources)

paths to follow. Currently, an aggregation tree is calculated for each independent event in the network, for each sink. This is an implementation detail however, and could be altered in future versions.

IV. Evaluation

In this section, we analyze the three data aggregation tree algorithms. The goal of this analysis is to determine the most energy-efficient aggregation tree given varying levels of aggregation efficiency.

We analyze the aggregation tree algorithms in terms of the amount of energy spent transmitting sensor data. Because the algorithms are implemented in the ns2 network simulator [14], a link energy model is required to provide the energy analysis. We begin by presenting this model. Next, the simulation set up is described. Finally, the performance analysis is presented.

A. Simulation Setup

Our simulations are performed with the ns2 network simulator. The sensor network consists of a 100 node network laid out on a grid. Each node in the middle of the grid has 8 one-hop neighbors. The sensor data size is 64 bytes and the packet header size is 6 bytes. Each node has a transmit power of 36mW and a receive power of 5.4mW. These values were chosen to model a common sensor node [18]. The data transmission rate of the nodes is 40Kbps.

We consider three sensor scenarios. The first scenario is where an event occurs at the corner of the sensor network (see Figure 2, nodes 1-5 are sources). This is a common model for data aggregation studies. The second scenario is where an event occurs in the middle of the network (see Figure 2, nodes A-E are sources). The final scenario is where events occur at the edges of the network (see Figure 2, nodes U-Z are sources).

In all scenarios, the sink is placed near the bottom edge of the network. For the simulations, no mobility is used since we assume a static sensor network. Additionally, we do not consider node failure.

B. Experimental Results

Two groups of results are presented in this section. The first group evaluates the performance of Oceanus, the opportunistic algorithm, the greedy incremental algorithm, and sending the data via shortest path routes for a perfect aggregation algorithm with no cost. The second group of results evaluates the performance of these methods across aggregation algorithms with varying aggregation efficiencies, but no aggregation cost. Varying the aggregation cost has the same effect. For all experiments presented here, we consider only the energy expenditure of nodes within the aggregation tree. This is reasonable because the focus of this paper is finding the minimum cost aggregation trees in terms of energy. Developing algorithms for putting nodes to sleep that are not part of the aggregation tree is outside the scope of this work.

1) Perfect Aggregation: The results in this section use a perfect aggregation function. There is no cost for aggregation and the amount of data sent after aggregation is 64 bytes, the same as a unit of sensor data, no matter how many flows are being aggregated. Therefore, the optimal aggregation trees aggregate flows as close to the sources as possible. These experiments explore the most efficient end of the aggregation spectrum. For each of the three algorithms tested, opportunistic, greedy incremental, and Oceanus, the algorithms are run over 10 varying networks conforming to the three basic configurations. The results presented are the average results from these runs. The graphs in this section are normalized to the energy expenditure in mJ of sending each sensor's data via a shortest path link with no aggregation. This yields percentage savings over the shortest path, no aggregation method for each of the algorithms.

Figure 3 depicts the energy savings for networks where the sensors are clustered in a corner of the network. Because the nodes are clustered, the opportunistic algorithm has a reasonable likelihood of causing two streams to flow to the sink only one hop away from each other. However, the greedy incremental algorithm is able to aggregate the flows somewhat near to the sources. Oceanus, on the other hand, links all of the nodes together in a chain , aggregating at the sources, and transmits through the closest node

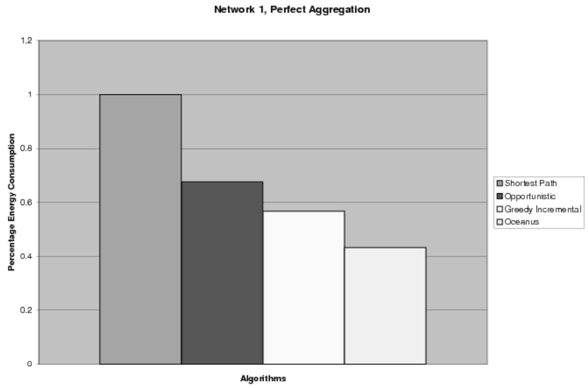

Fig. 3. Perfect Aggregation, Network 1

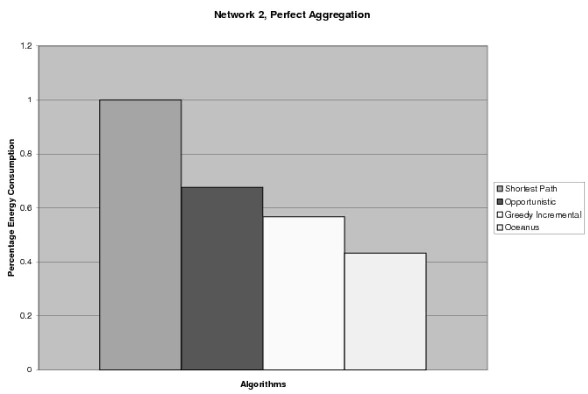

Fig. 4. Perfect Aggregation, Network 2

to the sink with high probability. Oceanus uses 26% less the energy then the baseline and about 15% less than the greedy incremental algorithm.

Figure 4 depicts the energy savings for networks where the sensors are surrounding an event in the middle of the network. In this scenario, it is likely that the nodes on the side of the event away from the sink will aggregate with nodes closest to the sink rather early. For the greedy incremental algorithm, it is more likely that the nodes aggregate close to the sources. Oceanus sends data around the event in a ring, aggregating at each source, and then connects the ring to the sink via one of the source nodes. Oceanus uses 40% less energy then the baseline and about 10% less energy than the greedy incremental algorithm.

Figure 5 depicts the energy savings for networks where the sensors are on different edges of the network. This scenario represents a difficult case. With high probability, the opportunistic algorithm can only aggregate those

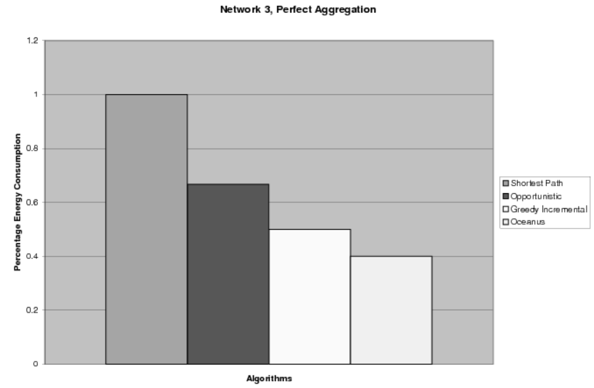

Fig. 5. Perfect Aggregation, Network 3

nodes in each sector of the network. Therefore, it is likely that multiple independent streams will be sent through the network without aggregation. The greedy incremental algorithm will likely send all data across the center of the network, aggregating in the middle, depending on which stream begins first. Oceanus routes data around the edges of the network, in a large circle, from source node to source node, aggregating at the sources. Oceanus uses 43% less energy than the baseline and about 10% less energy than the greedy incremental algorithm. This scenario is the worst case for the opportunistic algorithm. As expected, the greedy incremental method and Oceanus perform well, with Oceanus providing greater energy efficiency by routing through all of the source nodes. Therefore, for perfect aggregation functions, Oceanus outperforms all other aggregation methods.

The total energy consumed for varying lengths of data flows from 1 sensor data packet from each sensor to 100 packets from each sensor was also tested for perfect aggregation. The longer the flows go on, the cost of setting up the aggregation tree is amortized over the length of the flow. For each of the three algorithms tested, opportunistic, greedy incremental, and Oceanus, we ran the algorithms over 10 varying networks conforming to the three basic configurations. This showed the trends in the data over a range of length of flows. Since the none of the aggregation trees can be set up before the first packets are received, all of the algorithms have the same energy consumption as the Shortest Path flows. However, as the flows progress, each of the algorithms' energy consumptions diverge. As expected, the results remained the same as presented in Figures 3-5, with the gaps between the tree algorithms increasing roughly linearly as the flow lengths increased.

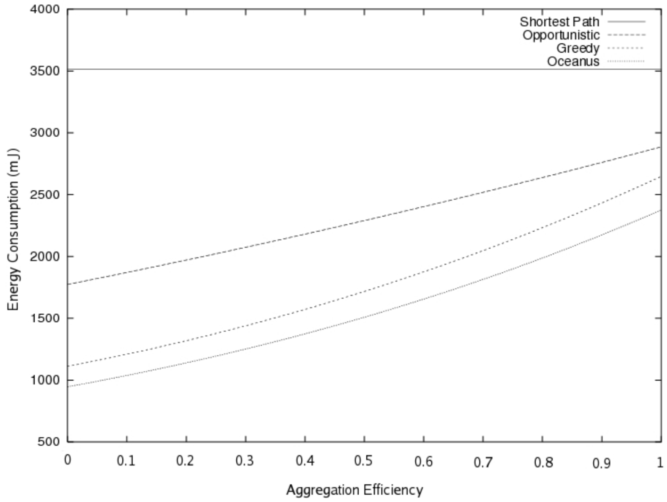

Fig. 6. Energy Consumption vs. Aggregation Efficiency, Network 1

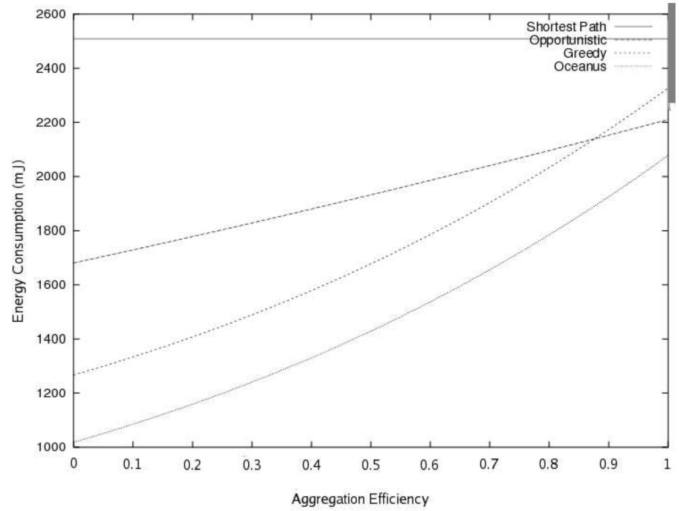

Fig. 7. Energy Consumption vs. Aggregation Efficiency, Network 2

2) Varying the Aggregation Efficiency: The locations of the optimal aggregation points depends on the efficiency of the aggregation algorithm. Oceanus attained the most efficient communication for perfect aggregation functions by aggregating data close to the source nodes. However, as the efficiency of the aggregation algorithm decreases, the optimal aggregation points move closer to the sink node.

The graphs in this section present the amount of energy to send 100 sensor packets from each sensor node to the sink using the three aggregation methods (opportunistic, greedy incremental, and Oceanus) as well as shortest path routing. The x-axis of the graphs represent the amount of data compression achieved by the aggregation function. Zero represents a perfect aggregation function and one represents simple concatenation.

Figure 6 depicts the energy consumed for varying aggregation efficiencies in a network where the sources are in the corner of the network. The opportunistic algorithm curve is rather linear, as expected. This is because the data aggregation is performed only if shortest paths from the source nodes cross. Therefore, it is affected least by changes in the aggregation efficiency. Both the greedy incremental method and Oceanus begin to perform more poorly as the aggregation efficiency decreases. This is because the benefit of aggregation shrinks below the possible increase in hops a flow makes to reach an aggregation point. However, because the sources are collected in a corner of the network, no path is lengthened significantly. Therefore, both the greedy incremental algorithm and Oceanus continue to outperform the opportunistic by a significant margin, with Oceanus always being the most efficient.

Figure 7 depicts the energy consumed for varying aggregation efficiencies in a network where the sources are circled around an event in the middle of the network. Again, the curve for the opportunistic method is roughly linear as expected and both the greedy incremental algorithm and Oceanus begin to suffer as the aggregation algorithm becomes less efficient. This time however, the greedy incremental algorithm begins to consume more energy than the opportunistic algorithm for efficiencies close to simple concatenation. This is because some paths are lengthened to reach aggregation points, but the gain in aggregation no longer outweighs these path increases. Oceanus continues to be the most efficient algorithm in this case as well. This is because its gains in aggregation follow along the shortest links to the source node and therefore still outweigh any increases in path length.

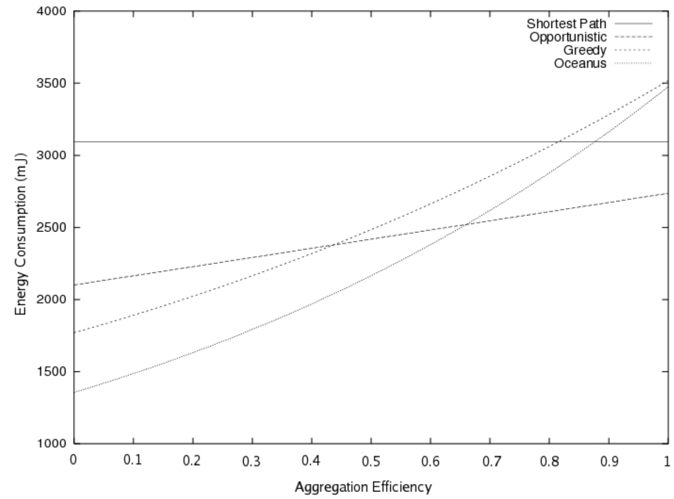

Fig. 8. Energy Consumption vs. Aggregation Efficiency, Network 3

Figure 8 depicts the energy consumed for varying aggregation efficiencies in a network where the sources are around the sides of the network. As the graph shows, it is the worst case for Oceanus at poor aggregation efficiencies. Around 82% aggregation efficiency, the opportunistic algorithm begins to outperform Oceanus. This is because, the Oceanus algorithm always tries to perform some aggregation, but in this case, this causes poor performance for the worst aggregation efficiencies. However, Oceanus continuously outperforms the greedy incremental algorithm.

3) Summary: Oceanus significantly outperforms both the opportunistic method as well as the greedy incremental method of aggregation for perfect aggregation functions. Furthermore, in all three network scenarios, Oceanus outperforms the greedy incremental method for all aggregation function efficiencies. However, for aggregation efficiencies in the worst 18%, the opportunistic method outperforms Oceanus in the scenarios where the sources are scattered on all sides of the network. However, this scenario is unlikely, since it is rare that data from events occurring in completely different areas of the network will be aggregated. Therefore, in most realistic scenarios, Oceanus outperforms both the opportunistic method and the greedy incremental method of data aggregation. This is because of Oceanus takes into account the energy efficiency of the aggregation algorithm in creating its aggregation trees. Finding efficient data aggregation algorithms is itself the topic of other work; however, the more efficient the aggregation algorithms, the more efficiently data can be transmitted through a sensor network.

V. Conclusions and Future Work

This paper has explored the aggregation efficiency space and where aggregation points should be located within a sensor network to provide energy efficient communication. We have demonstrated that for high energy efficient aggregation algorithms, aggregation points should lie close to the sources of the data. However, as aggregation efficiency decreases, aggregation points should be migrated towards the sink. Therefore, we present Oceanus, which builds the aggregation trees based on the efficiency of the aggregation algorithm. We have shown that Oceanus outperforms both the opportunistic and the greedy incremental methods over a range of network topologies and aggregation efficiencies.

This analysis suggests a future direction for aggregation tree algorithm design. A distributed algorithm that migrates the aggregation points towards the source for high-efficiency aggregation algorithms and towards the sink for low efficiency algorithms is a future direction. Future work also consists of finding energy efficient means of taking care of failures in the network and considering adding load-balancing. We have not implemented any load balancing mechanism, or fault tolerance into Oceanus.

References

[1] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Mobile Data Management*, 2001.

[2] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad-hoc heterogeneous sensor networks. In *Intl J. High Performance Computing Applications*, 2002.

[3] Qing Fang, Feng Zhao, and Leonidas Guibas. Lightweight sensing and communication protocols for target enumeration and aggregation. In *MobiHoc*, 2003.

[4] M. Garey and D. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, 1979.

[5] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The sensor network as a database. Technical Report 02-771, Computer Science Department, University of Southern California, September 2002.

[6] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, 2000.

[7] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS*, 2002.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *MobiCom*, 2000.

[9] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *DEBS*, 2002.

[10] Rajnish Kumar, Matthew Wolenetz, Bikash Agarwalla, JunSuk Shin, Phillip Hutto, Arnab Paul, and Umakishore Ramachandran. Dfuse: A framework for distributed data fusion. In *Proceeding of the First International Conference on Embedded Network Sensor Systems*, 2003.

[11] Stephanie Lindsey, Cauligi Raghavendra, and Krishna Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9), 2002.

[12] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[13] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. The design fo an acquisitional query processor for sensor networks. In *SIGMOD*, 2003.

[14] ns2 Network Simulator. http://www.isi.edu/nsnam/ns/.

[15] Younis Ossama and Sonia Gahmy. Heed: A hybrid, energy-efficient, distributed clustering apporach for ad-hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4), 2004.

[16] Narayanan Sadagopan, Bhaskar Krishnamachari, and Ahmed Helmy. The aquire mechanism for efficient querying in sensor networks. In *SNPA*, 2003.

[17] C. Sengul and R. Kravets. Conserving energy with on-demand topology management. In *Second IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, November 2005.

[18] XBOW. 2nd generation micamote. http://www.xbow.com.

[19] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. In *SIGMOD*, 2002.