Impact of Network Density on Data Aggregation in Wireless Sensor Networks *

Chalermek Intanagonwiwat, Deborah Estrin †, Ramesh Govindan ‡,
John Heidemann

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292, USA
{intanago, estrin, govindan, johnh}@isi.edu

November 4, 2001

Abstract

In-network data aggregation is essential for wireless sensor networks where resources (*e.g.,* bandwidth, energy) are limited. In a previously proposed data dissemination scheme, data is opportunistically aggregated at the intermediate nodes on a low-latency tree which may not necessarily be energy efficient. A more energy-efficient tree is a greedy tree which can be incrementally constructed by connecting each source to the closest point of the existing tree. In this paper, we propose a greedy approach for constructing a greedy aggregation tree to improve path sharing. We evaluated the performance of this greedy approach by comparing it to the prior opportunistic approach. Our preliminary result suggests that although the greedy aggregation and the opportunistic aggregation are roughly equivalent at low-density networks, the greedy aggregation can achieve signficant energy savings at higher densities. In one experiment we found that the greedy aggregation can achieve up to 45% energy savings over the opportunistic aggregation without an adverse impact on latency or robustness.

1 Introduction

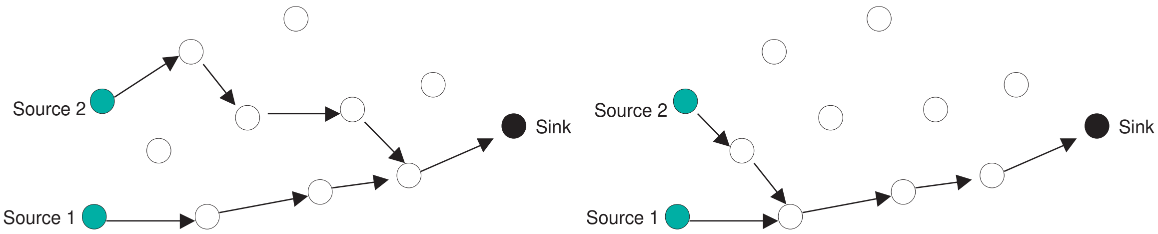
Advances in radio, sensor, and VLSI technology will enable small and inexpensive sensor nodes capable of wireless communication and significant computation. Large-scale networks of such sensors may require novel data dissemination paradigms which are scalable, robust, and energy-efficient [7]. One such paradigm is directed diffusion [12] which incorporates data-centric routing and application-specific processing inside the network (*e.g.,* data aggregation). Given that the communication cost is several orders of magnitude higher than the computation cost [16], directed diffusion can achieve significant energy savings with in-network data aggregation. This benefit of data aggregation has been confirmed theoretically [14] and experimentally [10].

The instantiation of directed diffusion described in the earlier work establishes low-latency paths between sources (sensor nodes that detect phenomena) and sinks (user nodes) using only localized algorithms. Data from different sources can be *opportunistically* aggregated at intermediate nodes along the established paths. Energy-wise,

* This work was supported by the Defense Advanced Research Projects Agency under grant DABT63-99-1-0011. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

† University of California, Los Angeles and USC/Information Sciences Institute

‡ International Computer Science Institute and USC/Information Sciences Institute



(a) Late Aggregation **(b)** Early Aggregation

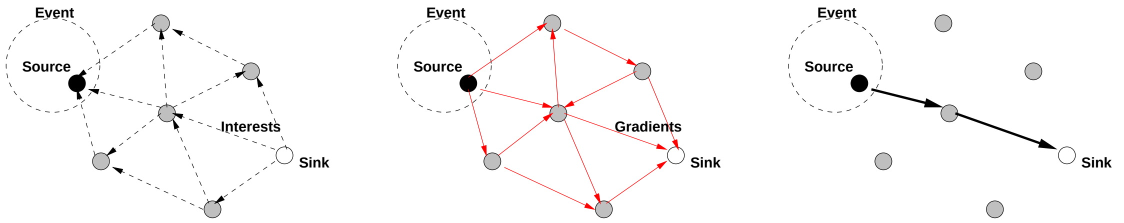
Figure 1: An Example of Late Aggregation and Early Aggregation.

such path selection with opportunistic aggregation is not optimal because paths from different sources to a sink may not be early shared or merged. Thus, data may not be aggregated (or reduced) near the sources (Figure 1). To achieve more energy savings, one could design another diffusion instantiation which favors path selection to increase early sharing of paths and reduce energy consumpition.

However, there are drawbacks for early path sharing: higher latency and less robustness. Some paths will be longer to trade off for early aggregation. Given that an aggregate generally contains more information than a non-aggregated message, a loss of an aggregate adversely impacts the quality of the result more than a loss of a non-aggregated message. Conversely, there are practical constraints that favor early path sharing. For example, the early aggregation reduces overall traffic which is preferable, given the limited bandwidth (Section 5).

Nevertheless, it is not obvious that such path optimization will practically lead to significant energy savings. Specifically, the additional cost of establishing the optimal aggregation tree could be dramatically high. Assuming perfect aggregation (*i.e.,* the data size of an aggregate is equal to the data size of an individual event), the cost of the tree is the number of links on the tree. Therefore, finding the optimal aggregation tree is computationally infeasible because it is equivalent to finding the Steiner tree that is known to be NP-hard [23]. Thus, we do not expect to get an optimal tree. Imperfect aggregation and a sub-optimal aggregation tree can further limit the energy savings.

One promising heuristic is a greedy incremental tree (GIT) [18]. To construct a greedy incremental tree, a shortest path is established for only the first source to the sink whereas each of the other sources is incrementally connected at the closest point on the existing tree. In this paper, we propose a new instantiation of directed diffusion that uses a GIT-like algorithm (Section 4) to improve path sharing. We have implemented this *greedy-tree* approach in ns-2 and we compare it to our prior *opportunistic* approach (Section 5). Our preliminary result suggests that although the two are roughly equivalent at low-density networks, the greedy-tree can achieve signficant energy savings at higher densities. Recent work has compared the greedy incremental tree with the shortest path tree (SPT) using *abstract simulations* [14]. Based on the event-radius model and the random sources model, their results indicate that the transmission savings by the GIT over the SPT do not exceed 20%. However, the energy savings of our greedy aggregation can definitely be *much higher than 20%*, given our source placement schemes and high-density networks (Section 5.4).



(a) Interest propagation

(b) Initial gradients set up

(c) Data delivery along reinforced path

Figure 2: A simplified schematic for directed diffusion.

2 Directed Diffusion

Directed diffusion [12] is an approach to attribute-based data communication for wireless sensor networks. The goal of directed diffusion is to establish efficient communication between sources and sinks. Directed diffusion consists of several elements. Data is *named* using attribute-value pairs. A sensing task (or a subtask thereof) is disseminated throughout the sensor network as an *interest* for named data. This dissemination sets up *gradients* within the network designed to "draw" events (*i.e.,* data matching the interest). Events start flowing towards the originators of interests along multiple paths. The sensor network *reinforces* one, or a small number of these paths. Figure 2 illustrates these elements.

To describe the elements of diffusion, we take the simple example of a sensor network designed for tracking animals in a wilderness refuge. Suppose that a user in this network would like to track the movement of animals in some remote sub-region of the park. In directed diffusion, this tracking task represents an *interest*. An interest is a list of attribute-value pairs that describe a task using some task-specific naming scheme. Intuitively, attributes describe the data that is desired by specifying sensor types and some geographic region. They are then used to identify and contact all relevant sensors. We use the term *sink* to denote the node that originates an interest.

The interest is propagated from neighbor-to-neighbor towards sensor nodes in the specified region. A key feature of directed diffusion is *that every sensor node is task-aware*—by this we mean that nodes store and interpret interests, rather than simply forwarding them along. In our example, each sensor node that receives an interest remembers which neighbor or neighbors sent it an interest. To each such neighbor, it sets up a *gradient*. A gradient represents both the direction towards which data matching an interest flows, and the status of that demand (whether it is active or inactive and possibly the desired update rate). Subsequent to setting up a gradient, the sensor node redistributes the interest to its neighbors. To do this, the node floods the interest to all its neighbors, or send only to a subset of neighbors in the direction of the specified region.

When a sensor node that matches the interest is found, it activates its local sensors to begin collecting data. (Prior to this, the node's sensing circuitry operates in a low-power mode). The sensor node then generates *data* messages matching the interest. In directed diffusion, data is also represented using an attribute-value pair based naming scheme. A sensor node that generates such an event description is termed a *source*.

The source sends the data message to each of those neighbors for which it has gradients for the matching interest.

A node that receives this message checks if it has received the identical message before—for this purpose, each node maintains a small *cache* of recently received data items. This cache serves to avoid duplicates, prevent loops, and can be used to preferentially forward interests. This is a key capability of directed diffusion. If an identical data item exists in the cache, the node drops the message. The node also triggers application-specific filters that do more sophisticated processing. Application-specific, in-network processing is another key feature of diffusion; in-network aggregation can reduce data transferred across the network, and thereby conserve energy. If this data item does not exist in its cache, the node determines the matching interest, and resends the data along the gradient towards the neighbor.

If the sink has multiple neighbors, it chooses to receive subsequent data messages for the same interest from a preferred neighbor (for example, the one which delivered the first copy of the data message). To do this, the sink *reinforces* the preferred neighbor, which, in turn reinforces its preferred upstream neighbor, and so on. Finally, if a node on this preferred path fails, sensor nodes can attempt to *locally repair* the failed path.

Even this simplified description points out several key features of diffusion, and how it differs from traditional networking. First, diffusion is data-centric; all communication in a diffusion-based sensor network uses interests to specify named data. Second, all communication in diffusion is neighbor-to-neighbor or hop-by-hop, unlike traditional data networks with end-to-end communication. Every node is an "end" in a sensor network. A corollary to this previous observation is that there are no "routers" in a sensor network. Each sensor node can interpret data and interest messages. This design choice is justified by the task-specificity of sensor networks. Sensor networks are not general-purpose communication networks. Third, nodes do not need to have globally unique identifiers or globally unique addresses. Nodes, however, do need to distinguish between neighbors. Finally, because individual nodes can cache, aggregate, and more generally, process messages, it is possible to perform coordinated sensing close to the sensed phenomena. It is also possible to perform in-network data reduction, thereby resulting in significant energy savings.

3 Data Aggregation and Directed Diffusion

Once sensors detect phenomena, generated events are disseminated to users. Intermediate nodes may aggregate several events into a single event to reduce transmissions and total data size for system resource savings. The total size reduction mainly depends on data characteristics, event representations, and applications. Only data aggregation that leads to total size reduction is considered and justified in this paper. Furthermore, data aggregation will also reduce transmissions. As a result, the total per-transmission overhead (*e.g.,* packet headers, MAC control packets) will be reduced and the energy savings will be even more evident.

Similar to data compression, data aggregation can be classified into two approaches (*i.e.,* lossless and lossy). With *lossless aggregation*, all detailed information is preserved. However, according to information theory, the total size reduction is upper bounded by entropy (*i.e.,* a measure of how much information is encoded in a message). The basic principle of data reduction is to eliminate redundant information. Given that events may be highly correlated, there will be a significant amount of redundancy among them. Unlike lossless aggregation, *lossy aggregation* may discard some detailed information and/or degrade data quality for more energy savings.

Examples of lossless aggregation are the timestamp aggregation and the packing aggregation. The timestamp aggregation can be used in a remote surveillance application where an event consists of several attributes including a timestamp. Distinct events may actually be temporally correlated (within seconds of one another). The redundant information (*e.g.,* the hour and minute field in the timestamp) may be minimized (*i.e.,* not repeated) using an efficient representation for aggregated events. For the packing aggregation, several non-aggregated messages are packed into one aggregate without compression. The only savings are the total per-transmission overhead, such as packet headers.

An example of lossy aggregation is the outline aggregation used in eScan [24]. The goal of eScan is to depict the remaining energy levels of sensor nodes. Leveraging the spatial locality of energy usage, topologically adjacent nodes can sometimes be approximately represented by a bounding polygon. This approach trades off some inaccuracy in node energy representation for reduced energy usage.

Directed diffusion was designed with application-level data processing in mind. Unsurprisingly, directed diffusion can take advantage of data aggregation due to in-network data processing capability. Nevertheless, some directed diffusion mechanisms can be adjusted to achieve even more benefit out of data aggregation. It is expected that, with the proper interactions between data aggregation rules and reinforcement rules, energy efficient paths would be selected rather than low delay paths. In particular, if aggregated data are distinguishable from non-aggregated data, it will be possible to design reinforcement rules that favor aggregated data paths over non-aggregated data paths. Those rules will encourage path sharing and achieve even more energy saving due to more data aggregation (See Section 4).

Energy savings of data aggregation depend on reduction in total data size. If the total data size is rarely reduced after aggregation, a shortest path will be more energy efficient than an aggregated data path. Moreover, without a reasonable reduction in total data size, aggregated data paths introduce traffic concentration (and probably also congestion) which adversely impacts network lifetime. Under this scenario, path optimization is not worth performing. Conversely, if the total data size is dramatically reduced after aggregation, it will be reasonable to trade off delay for energy efficiency by favoring longer but aggregated data paths over shorter but non-aggregated data paths. A *gradient map* (*i.e.,* data gradients from sources to sinks) similar to a greedy tree will be preferred. Specifically, aggregation points need to be carefully selected (using reinforcement) so that additional dissipated energy (due to longer paths) does not exceed energy savings (due to total data size reduction).

4 Greedy Aggregation

Greedy aggregation is a novel diffusion approach to construct a greedy incremental tree for data aggregation. This approach differs from the previous diffusion approach (*i.e.,* the opportunistic aggregation on a lowest latency tree) in path establishment and maintenance. To construct a greedy incremental tree, a shortest path is established for only the first source to the sink whereas each of the other sources is incrementally connected at the closest point on the existing tree.

4.1 Path Establishment

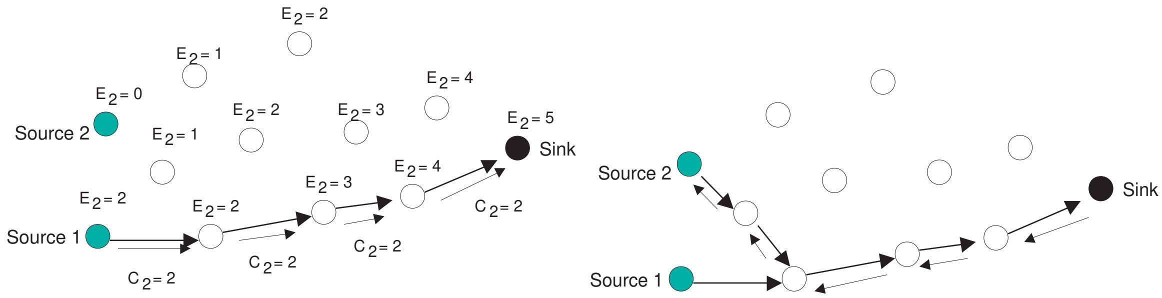
For the opportunistic aggregation, the sink initially diffuses an interest for a low event-rate notification (*i.e.,* an *exploratory* event) intended for path establishment and repair. We call the gradients set up for sending exploratory events *exploratory gradients*. Once a source detects a matching target, it sends exploratory events, possibly along multiple paths, toward the sink. After the sink receives these exploratory events, it *reinforces* one particular neighbor in order to "draw down" real data (*i.e.,* events at a higher data rate that allow high quality tracking of targets). We call the gradients set up for sending high quality tracking events *data gradients*. The local rule for selecting an *empirically low delay path* is to reinforce any neighbor from which a node recieves a previously unseen exploratory event.

For our greedy aggregation, each event contains an additional attribute E, the energy cost for delivering this event from the source to the current node. Once a source detects phenomena, it sends exploratory events with $E = 1$ to each neighbor for whom it has a gradient. After a node receives these previously unseen exploratory events, it adds the cost of that transmission to E before resending. Since our radios have fixed transmission power, we measure energy as equivalent to hops, but direct measures of variable energy could also be used. Although this energy attribute is useful for selecting a lowest-energy path, it is not sufficient for constructing a greedy incremental tree because there is no path-sharing information.

To provide such information, each source on the existing tree (*i.e.,* a source with data gradients) also generates an *incremental cost message*, once it receives a previously unseen exploratory message generated by other sources. The incremental cost message contains the random message id of the corresponding exploratory event and the additional energy cost C required for delivering that exploratory event to the existing tree. This incremental cost message is only sent along the existing tree using data gradients. Unlike the energy cost E, the incremental energy cost C can only be decreased (in order to find the closest point on the existing tree). Once an on-tree node receives a previously unseen incremental cost message, it searches for the corresponding exploratory event in its message cache. The node updates C of the incremental cost message to the minimum value between the current C and the energy cost E' of the exploratory event retrieved from the cache, before sending the incremental cost message to its outgoing data gradients.

Once a sink receives a previously unseen exploratory event, it does not reinforce a neighbor immediately because an energy-efficient path is not necessarily a lowest-delay path. Instead, a reinforcement timer of T_p is set up. After the timer expires, the sink reinforces any neighboring node that sent the exploratory event or the incremental cost message (of the corresponding exploratory event) at the lowest energy cost. If the energy cost of an exploratory event and the incremental cost message are equivalent, the sink reinforces the neigbhoring node that sent the exploratory event. Other ties are decided in favor of the lowest delay. When the neighboring node receives the reinforcement message (associated with the random message id of the exploratory message), the node sets a data gradient towards the sink (or the node that sends the reinforcement) and reinforces an upstream neighboring node immediately using the above local rule without setting up a timer.

As a result, a greedy incremental tree can be constructed using the described local rule (See Figure 3 for example). Unlike the exploratory event, the incremental cost message contains the minimum energy cost of delivering data from a new source to the existing tree (not to the sink). The path from the first source to the sink is a lowest energy path because of no existing tree or no incremental cost message genearted yet. Each subsequent source is incrementally connected to the aggregation tree at the closest point (using the energy cost information provided by the incremental cost message). Hence, one might wonder if this algorithm requires unsynchronized sources so that each source can be incrementally connected to the tree. We do not make that assumption because sources can be



(a) An Incremental Cost Message **(b)** Reinforcement

Figure 3: An Example of Path Establishment.

synchronized if they are triggered by the same phoenomena. Although we have described the algorithm using examples in which different sources start at different times, the algorithm in fact works when sources start transmitting events in near simultaneity. In that scenario, the algorithm initially constructs a lowest-energy-path tree (*i.e.,* each source is connected to the sink using the lowest-energy path), but this problem is not persistent. At the subsequent round of exploratory events, the greedy incremental tree will be constructed and the lowest-energy-path tree will be pruned off using the negative reinforcement mechanism in Section 4.3.

4.2 Data Aggregation and Set Covering Problem

To enable data aggregation, intermediate nodes will process or delay received data for a period of time T_a before sending them. This delay is crucial for data aggregation because multiple data will not be received at the same time. The delay should be selected based on the application or other system factors. For example, in a TDMA MAC, one might match the aggregation time to a multiple of the TDMA frame duration, or as a fraction of the periodicity of sensor data generation. Thus, unsent data can be periodically aggregated and appropriately delivered to neighbors using data gradients. However, intermediate nodes do not necessarily delay received data for the same period of time. An intermediate node that receives a sufficient amount of data for aggregation does not need to delay the received data any further. In addition, an intermediate node that is not an aggregation point does not need to delay the data at all.

Similar to exploratory events, messages and aggregates also contain the energy cost attribute. After aggregating multiple messages into an aggregate, the final step before sending the aggregate is to compute the associated energy cost for that aggregate. This energy information will be used for empirical adaptation to energy-efficient paths. Specifically, the negative reinforcement rule uses this information for path truncation (Section 4.3). However, different neighbors might report aggregates of different subsets of data items, with varying costs. The challenge is to find the set of incoming aggregates which cover the data items at the smallest cost. Given incoming aggregates, it is not trivial to calculate the minimum energy cost of the outgoing aggregate because it is a *weighted set-covering problem* [4, 6] (*i.e.,* a generalized version of an NP-hard set-covering problem).

An instance of the set-covering problem consists of a finite set $X = \{x_1, x_2, ..., x_m\}$ and a family F of subsets

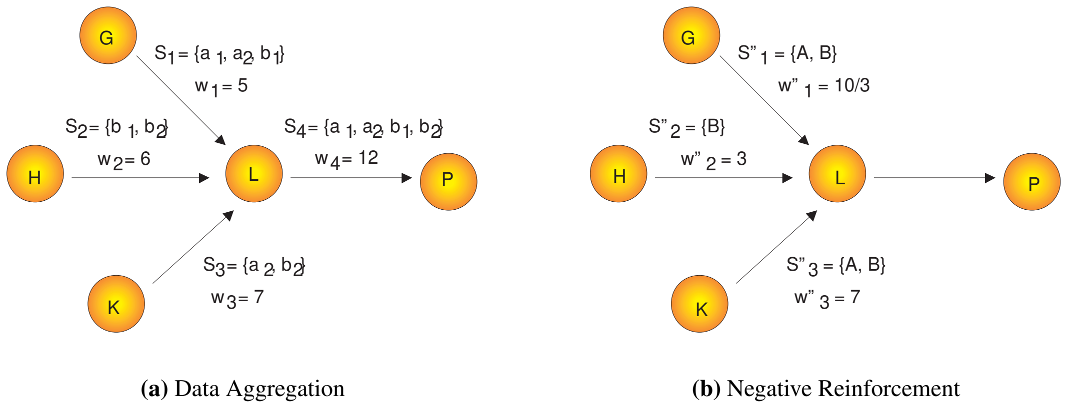


Figure 4: The Use of Weighted Set Covering Problems.

S_i of X, $F = \{S_1, S_2, ..., S_n\}$, such that every element of X belongs to at least one subset in F: $X = \cup_{i=1}^{n} S_i$. The regular set-covering problem is to determine a minimum-size subset $C \subseteq F$ whose members cover all of X, *i.e.,* $X = \cup_{S_i \in C} S_i$. For the weighted set-covering problem, each set S_i in the family F is associated with a weight w_i and the weight of a cover C is $\sum_{S_i \in C} w_i$. The problem is to find a minimum-weight cover (The regular set-covering problem is a special case of the weighted set-covering problem with $w_i = 1$ for all i).

Our problem can be directly mapped into the weighted set-covering problem as follows. An incoming aggregate is a subset S_i whereas the outgoing aggregate is X. Each incoming aggregate is associated with the energy cost w_i. Therefore, the energy cost of the outgoing aggregate is the minimum weight of the cover plus 1.

Approximate algorithms for this problem include greedy heuristics [4, 6], probabilistic methods [17], genetic-algorithms-based heuristics [15], neural-networks-based techniques [9], and Lagrangian heuristics [5]. We chose the greedy heuristic because of its high-quality solutions (The worst ratio between the cost of a greedy solution and the optimal solution is $\ln d + 1$ where d is the maximum size of any set S_i [6, 9]). The heuristic of the greedy set-covering algorithm is to greedily select the next subset (among the remaining subsets) for covering uncovered elements at the lowest cost ratio until all elements are covered. The cost ratio r_i of S_i is $w_i/|S_i'|$ where $S_i' \subset S_i$ is the set of uncovered elements in S_i. However, there might exist a subset S_k in C where all elements in S_k are covered by the union of other subsets in C. The final step of the greedy heuristic is to remove such redundant subsets from C.

For example (Figure 4(a)), node **L** receives incoming aggregates $S_1 = \{a_1, a_2, b_1\}, S_2 = \{b_1, b_2\}$, and $S_3 = \{a_2, b_2\}$. Suppose that $w_1 = 5, w_2 = 6$, and $w_3 = 7$ are the associated energy costs. Therefore, $r_1 = 5/3, r_2 = 6/2$, and $r_3 = 7/2$ are the initial cost ratios. Given that r_1 is the minimum cost ratio, S_1 is the first selected subset and the remaining uncovered element is b_2. At the second step, $r_2 = 6/1$ and $r_3 = 7/1$ because $S_2' = \{b_2\}$ and $S_3' = \{b_2\}$. Thus, S_2 is selected as the final subset of the cover. **L** then sends an outgoing aggregate $S_4 = S_1 \cup S_2 = \{a_1, a_2, b_1, b_2\}$ to **P** with associated energy cost $w_4 = w_1 + w_2 + 1 = 12$.

4.3 Path Truncation

The algorithm described in Section 4.1 can result in more than one path being reinforced after multiple rounds of exploratory events (due to synchronized sources and network dynamics). For energy efficiency, we need a mechanism to *negatively reinforce* (to truncate) unnecessary or inefficient paths.

To find the inefficient paths, our trucation rule might also need to compute the set cover. Given that different neighbors might report aggregates of different subsets of data items, with varying costs, the challenge is to find the set of neighbors who cover the data items at the smallest cost. A simple truncation rule is to negatively reinforce neighbors that are not in the set cover.

Specifically, one plausible choice for a truncation rule is to negatively reinforce neighbors from which no energy-efficent aggregates have been received within a window of N events or time T_n. The local rule we evaluate in Section 5 is based on a time window of T_n, chosen to be 2 seconds in our simulations (or 4 times of the aggregation delay T_a). An incoming aggregate is considered energy-efficient if it is selected as a subset in the set cover C. For example (Figure 4(a)), node **L** will negatively reinforce node **K** because S_3 is not in C. However, given that events a_i and b_i are generated by sources A and B, respectively, this direct approach is a bit conservative and energy inefficient (**G** will likely send b_2 in its next aggregate. Therefore, **H** should also be negatively reinforced).

Our more energy-efficient rule is to compute the set cover C of sources not events. To do this, each event in an aggregate is replaced by its source. To preserve the initial cost ratio, the new associated energy cost w_i'' of the transformed aggregate S_i'' is $w_i * |S_i''|/|S_i|$. For example, the subsets of events in Figure 4(a) can be transformed to the subsets of sources in Figure 4(b). After the transformation, $S_1'' = \{A, B\}, S_2'' = \{B\}$, and $S_3'' = \{A, B\}$ whereas $w_1'' = 5 * 2/3, w_2'' = 6 * 1/2$, and $w_3'' = 7 * 2/2$. The cost ratios are still $r_1 = 5/3, r_2 = 3$, and $r3 = 7/2$. Using the greedy heuristic, S_1'' is selected as the only subset in C. Therefore, **L** negatively reinforces **H** and **K**.

When **H** receives this negative reinforcement, it degrades its gradient towards **L** (from a data gradient to an exploratory gradient). Furthermore, if all its gradients are now exploratory, **H** negatively reinforces those neighbors that have been sending data to it (as opposed to exploratory events). This sequence of local interactions ensures that the path through **H** is degraded rapidly.

5 Performance Evaluation

In this section, we report on some results from a preliminary simulation. We use a packet-level simulation to explore (in some detail) the implications of some of our design choices. This section describes our methodology, compares the performance of the greedy aggregation against the opportunistic aggregation, then explores impact of network dynamics and other factors on simulation.

5.1 Methodology

We implemented our greedy aggregation in the *ns-2* simulator [3]. Our goals in conducting this evaluation study were four-fold: First, verify the viability of the greedy aggregation as an alternative instantiation of directed diffusion. Second, understand the impact of network dynamics (*e.g.,* node failures) on the greedy aggregation and the opportunistic aggregation. Third, explore the influence of the source placement scheme on the performance of both approaches. Finally, study the sensitivity of the greedy aggregation to the parameter choices.

We select three metrics to analyze the performance of the greedy aggregation and to compare it to the opportunistic aggregation: average dissipated energy, average delay, and distinct-event delivery ratio. These metrics were used in earlier work to compare diffusion with other idealized schemes [12]. **Average dissipated energy** measures the ratio of total dissipated energy *per node* in the network to the number of *distinct* events received by sinks. This metric computes the average work done by a node in delivering useful information to the sinks. The metric also indicates the overall lifetime of sensor nodes. **Average delay** measures the average one-way latency observed between transmitting an event and receiving it at each sink. This metric defines the temporal accuracy of the phenomena detection delivered by the sensor network. The **distinct-event delivery ratio** is the ratio of the number of distinct events received to the number originally sent. This metric indicates the robustness to network dynamics. We study these metrics as a function of sensor network density.

To completely specify our experimental methodology, we need to describe the sensor network generation procedure (*e.g.,* our choice of ratio parameters, our workload). In order to study the performance of the greedy aggregation as a function of network density, we generate a variety of sensor fields in a 200m by 200m square. In most of our experiments, we study seven different sensor fields, ranging from 50 to 350 nodes in increments of 50 nodes. Each sensor field generated by randomly placing the nodes in the square. Each node has a radio range of 40m. Thus, the radio density (expressed in terms of how many other nodes each node can hear on average) ranges from 6 to 43 neighbors. For each network size, our results are averaged over ten different generated fields.

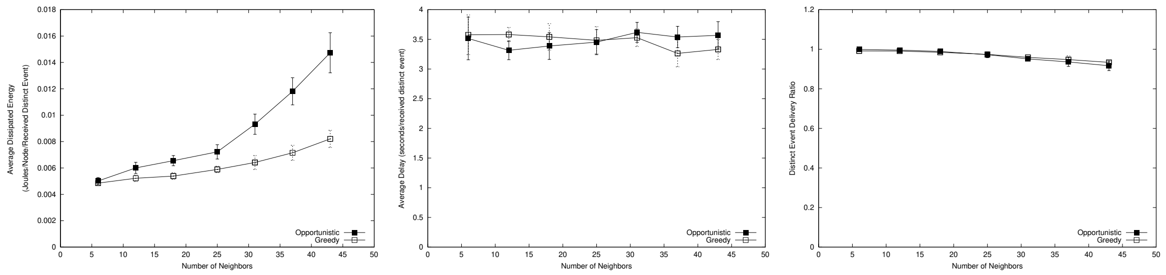
The *ns-2* simulator implements a 1.6 Mbps 802.11 MAC layer. Our simulations use a modified 802.11 MAC layer. To more closely mimic realistic sensor network radios [13], we altered the *ns-2* radio energy model such that the idle time power dissipation was about 35mW, or nearly 10% of its receive power dissipation (395mW), and about 5% of its transmit power dissipation (660mW).

Finally, in most of our simulations, we use a fixed workload which consists of five sources and one sink. All sources are randomly selected from nodes in a 80m by 80m square at the bottom left corner of the sensor field. The sink is randomly selected from nodes in a 36m by 36m square at the top right corner of the field. This source placement scheme differs from the event-radius model [14] for the low-level data aggregation because sources may not be triggered by the same phenomena and may not be within one hop from one another. Similar to the random source placement model [14], our source placement scheme is intended for the high-level data aggregation except that all sources in our placement scheme are placed in a subregion of the sensor field far from the sink. Our greedy aggregation targets high-level data (*e.g.,* abstract event represenation) because we expect that low-level data (*e.g.,* seimic signals) will be locally processed. Therefore, only the random source placement scheme and our source placement scheme are used in this evaluation.

Each source generates two events per second. Thus, the aggregation delay T_a is set to 0.5 seconds. The rate for exploratory events was chosen to be one event in 50 seconds. Events were modeled as 64 byte packets and other messages were 36 byte packets. The size of aggregates depends on the aggregation function and the number of data items in the aggregates (Section 5.4). For the perfect aggregation, the aggregate size is 64 bytes. Interests were periodically generated every 5 seconds and the gradient timeout was 15 seconds. We chose the window T_n for negative reinforcement to be 2 seconds and the timer T_p for positive reinforcement to be 1 second.

5.2 Comparative Evaluation

Our first experiment compares the greedy aggregation to the opportunistic aggregation for data dissemination in sensor networks. Figure 5(a) shows the average dissipated energy per packet as a function of the network density. Assuming the perfect aggregation, the greedy aggregation has noticeably better energy efficiency than the opportunistic approach for high-density networks. For some sensor fields, its dissipated energy is only 55% that of the opportunistic aggregation. Given a high-density network, there exist several shortest paths from a source to a sink. The available path diversity reduces the probability of path sharing among different sources for the opportunistic aggregation. The dissipated energy for both approaches increases with network size due to some diffusion overhead (*e.g.,* flooding of interests and exploratory events).



(a) Average dissipated energy **(b)** Average delay **(c)** Distinct-event delivery ratio

Figure 5: The greedy aggregation compared to the opportunistic aggregation.

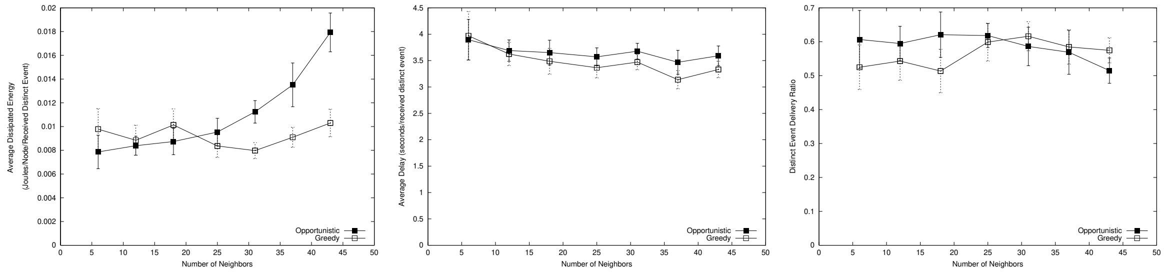
Figure 5(b) plots the average delay observed as a function of the network density. The greedy aggregation has a delay comparable to the opportunistic aggregation. This is a bit surprising because we expected that the greedy aggregation would trade off delay for energy efficiency. In low-density networks, the greedy aggregation has a bit longer delay because there are not many shortest paths. Thus, a shared path to a new source is unlikely to be a shortest path. Conversely, in high-density networks, the greedy approach has a bit lower delay because a shared path to a new source can still be close to a shortest path. Unlike the greedy aggregation, the opportunisitc approach does not encourage path sharing. Given the random jitter delay and the MAC fairness, the lowest-delay paths for sources are unlikely to be shared, particularly in high-density networks. Therefore, in high-density networks, the opportunistic aggregation is less energy-efficient and the overall traffic reduction is less. As a result, the delay of the opportunistic aggregation is a bit longer for the high-density networks, given a constant bandwidth.

The result in Figure 5(c) indicates that, for these uncongested networks without network dynamics, both approaches achieve the similar distinct-event delivery ratio as expected.

5.3 Impact of Network Dynamics

To study the impact of network dynamics on the greedy aggregation and the opportunistic aggregation, we simulated node failures as follows. For each sensor field, we repeatedly turned off 20% of nodes for 30 seconds. These nodes were uniformly chosen from the sensor field. Our dynamics experiment imposes fairly adverse conditions for a data dissemination protocol. At any instant, 20% of the nodes in the network are unusable. Furthermore, we do not permit any “settling time” between node failures.

As expected, node failures adversely impact the event delivery ratio of the greedy aggregation more than that of the opportunistic aggregation in the low-density networks (Figure 6(c)). With the network density increases, the size of the opportunistic tree increases due to less path sharing whereas the size of the greedy tree approximately remains the same. It is likely that there are more node failures on the opportunistic tree than the greedy tree. Therefore, the event delivery ratio of the greedy aggregation is higher than that of the opportunistic aggregation for high-density networks. This lower event delivery ratio of the opportunistic aggregation also results in its higher dissipated energy per event received (Figure 6(a)).



(a) Average dissipated energy **(b)** Average delay **(c)** Distinct-event delivery ratio

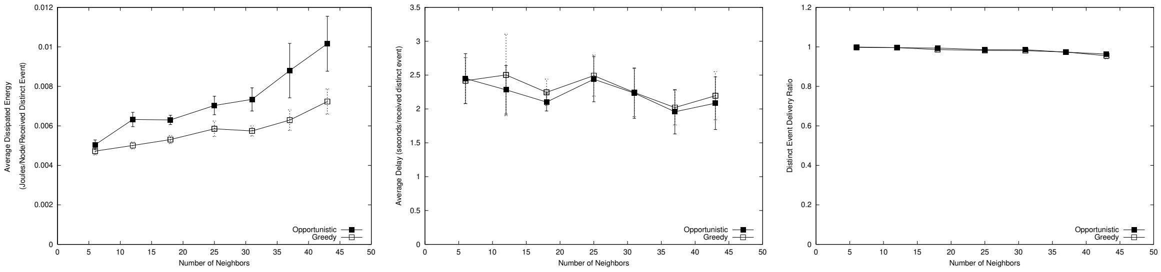
Figure 6: Impact of node failures.

5.4 Impact of Various Factors

To evaluate the sensitivity of our comparisons (Section 5.2), we re-ran our simulations using different parameters. In our previous comparisons, we placed 5 sources in the bottom left corner of the sensor field. How sensitive is our comparison to this source placement scheme ? In this experiment, we randomly placed 5 sources in the sensor field and re-ran our simulations of Section 5.2. Our results in Figure 7(a) indicate that the energy savings of the greedy aggregation are reduced to 30%. In these scenarios, sources are not necessarily closer to one another than to a sink. Even using the greedy incremental tree, the paths from the sources to the sink may not be early shared or merged.

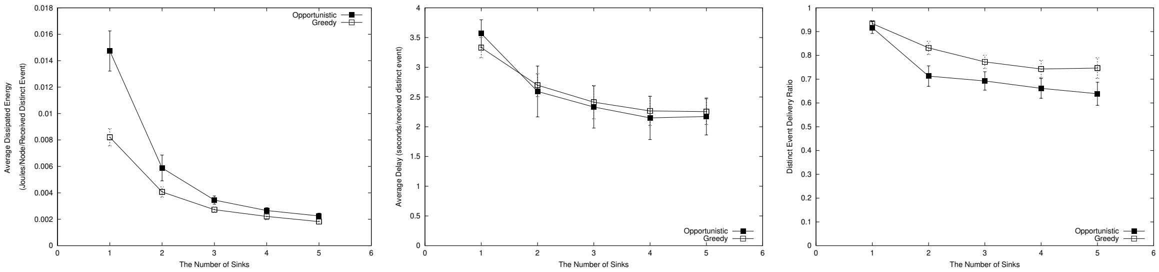
We also conducted an experiment to evaluate the sensitivity of our comparisons to the number of sinks (from 1 to 5 sinks) in the sensor field of 350 nodes. Similar to experiments in Section 5.2, 5 sources are randomly selected from nodes in a 80m by 80m square at the bottom left corner of the sensor field. The first sink is placed at the top right corner whereas the other sinks are uniformly scattered across the sensor field. With more sinks, the energy efficiency of the greedy aggregation is comparable to that of the opportunistic aggregation (Figure 8(a)). This impact of the random sink placement is similar to that of the random source placement. However, the event delivery ratio of the greedy aggregation is higher than that of the opportunistic aggregation because the early aggregation of the greedy aggregation reduces the overall traffic in general (Figure 8(c)).

To study the sensitivity of our comparisons to the number of sources, we re-ran our simulations in the sensor field of 350 nodes with 2, 5, 8, 11, and 14 sources. As the number of sources increases, the energy efficiency of



(a) Average dissipated energy **(b)** Average delay **(c)** Distinct-event delivery ratio

Figure 7: Impact of the random source placement.

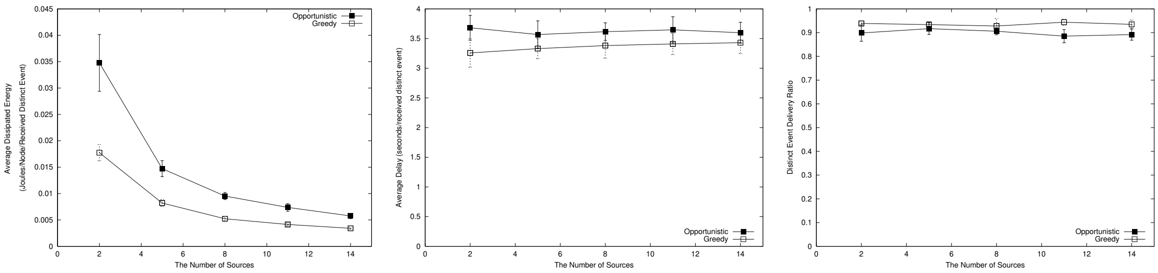


(a) Average dissipated energy **(b)** Average delay **(c)** Distinct-event delivery ratio

Figure 8: Impact of the number of sinks.

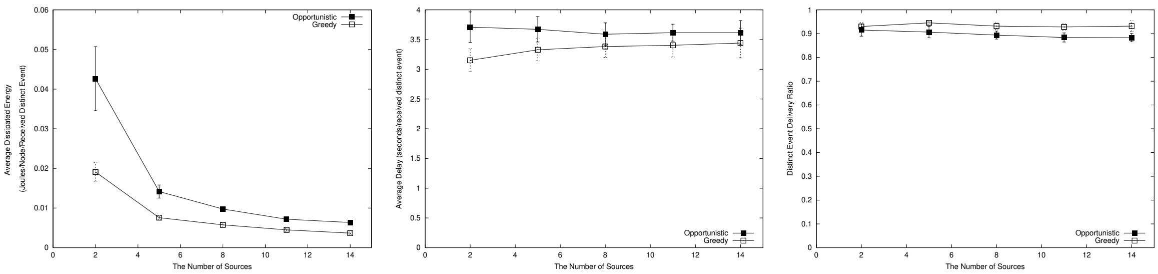
the greedy aggregation coverges to that of the opportunistic aggregation (Figure 9(a)). Given the high number of sources in the fixed area, our source placement scheme is approaching the event-radius model whereby sources are very close to one another. In these scenarios, paths from several sources are likely to be merged early even without path optimization.

So far, we assumed perfect aggregation for all our previous experiments. Of particular interest is the impact of other aggregation functions on the energy savings of the greedy aggregation. In this experiment, we used the linear aggregation for such sensitivity study. Given the linear aggregation, the size of aggregate (denoted by $z(S_i)$) is the linear function of data items in the aggregate S_i. Specifically, $z(S_i) = d_i * |x| + h$ where d_i is the number of data items in the aggregate, each data item size $|x|$ is 28 bytes, and the header size h is 36 bytes for our experiment. This linear aggregation is lossless but not energy-efficient because the only savings are the packet headers. Given that this aggregation function depends on the number of data items, we re-ran the previous experiment with this linear aggregation. As expected, the adverse impact of the inefficient aggregation function becomes more evident with the increased number of sources or data items (Figure 10(a)). In one experiment (11 sources), we found that the greedy aggregation can achieve 36% energy savings using the linear aggregation function whereas it can achieve 43% energy savings using the perfect aggregation function.



(a) Average dissipated energy **(b)** Average delay **(c)** Distinct-event delivery ratio

Figure 9: Impact of the number of sources.



(a) Average dissipated energy **(b)** Average delay **(c)** Distinct-event delivery ratio

Figure 10: Impact of the linear aggregation.

6 Related Work

Our work has been informed and influenced by a variety of other research efforts, which we now describe. Recent work in active networks [19, 22] and active services [1] has examined ways to provide in-network processing for the Internet. Sample applications include information transcoding, network monitoring, and caching. Their work targets Internet-like domains where communications are plentiful and computatonal power (expressed as CPU-cycles-per-packet) is limited. Conversely, our work targets sensor networks where communications are expensive and computational power is comparatively plentiful.

Another applicaiton of active services is Gathercast [2]. The goal of Gathercast is to losslessly aggregate small packets (with the same destination) into a larger packet for reducing the overhead of routing table lookups (not for reducing the total data size). Their work opportunistically aggregates small packets without altering any routing mechanisms or paths. Our work optimizes the aggregation tree for more energy savings.

Considered the reverse of a multicast tree, our aggregation tree has been inspired by some research efforts on the multicast tree construction. A greedy multicast tree [18] is probably the most energy efficient tree used by existing multicast protocols. The greedy multicast tree has been shown to outperform the shortest path tree in various studies [20, 21]. QoSMIC [8] constructs its greedy tree using the local search and offers alternate paths for

QoS requirements. Our greedy aggregation differently constructs the reverse tree using directed diffusion primitives coupled with a greedy heuristic for an NP-hard weighted set-covering problem.

In addition, recent efforts have pointed out some of the advantages of data aggregation in the context of sensor networks. Particularly, LEACH [11] can achieve energy savings by processing application-level data at its cluster heads whereas our approach can process such data along the greedy tree.

7 Conclusions

In this paper, we described a novel instantiation of directed diffusion (the greedy aggregation) that uses a GIT-like algorithm to improve path sharing with the low additional cost. Our greedy approach constructs an energy-efficient aggregation tree using data-centric reinforcement mechanisms (based on a greedy heuristic for weighted set-covering problems). We evaluated the performance of this greedy-tree approach by comparing it to our prior opportunistic approach. Our preliminary result suggests that although the greedy aggregation and the opportunistic aggregation are roughly equivalent at low-density networks, the greedy-tree approach can achieve signficant energy savings at higher densities. In one experiment we found that the greedy aggregation can achieve up to 45% energy savings over the opportunistic aggregation without an adverse impact on latency or robustness. Given that the energy is the scarce resource, this path optimization technique is essential for prolonging the lifetime of the highly-dense sensor networks.

References

[1] Elan Amir, Steven McCanne, and Randy H. Katz. An active service framework and its application to real-time multimedia transcoding. In *Proceedings of the ACM SIGCOMM*, pages 178–189, Vancouver, Canada, September 1998. ACM.

[2] B. Badrinath and P. Sudame. Gathercast: The design and implementation of a programmable aggregation mechanism for the internet. In *IEEE International Conference on Computer Communications and Networks (ICCCN)*, October 2000.

[3] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving simulation for network research. Technical Report 99-702b, University of Southern California, March 1999. revised September 1999, to appear in IEEE Computer.

[4] Thomas Bäck, Martin Schütz, and Sami Khuri. A comparative study of a penalty function, a repair heuristic, and stochastic operators with the set-covering problem. In *Artificial Evolution*, pages 3–20. Springer, Berlin, 1996.

[5] J.E. Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics*, 37:151–164, 1990.

[6] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[7] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'99)*, Seattle, Washington, August 1999.

[8] M. Faloustsos, A. Banerjea, and R. Pankaj. Qosmic: Quality of service multicast internet protocol. In *Proceedings of the ACM SIGCOMM*, September 1998.

[9] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *Euro. J. Operational Research*, 101(1):81–92, August 1997.

[10] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

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

[12] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'2000)*, Boston, Massachusetts, August 2000.

[13] William J. Kaiser. WINS NG 1.0 Transceiver Power Dissipation Specifications. Sensoria Corp.

[14] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. Modelling data-centric routing in wireless sensor networks. submitted for publication.

[15] G.E. Liepins, M.R. Hilliard, J. Richardson, and M. Palmer. Genetic algorithms applications to set covering and traveling salesman problems. In Donald E. Brown and Chelsea C. White III, editors, *Operations Research and Artificial Intelligence: The Integration of Problem-Solving Strategies*, pages 29–57. Kluwer Academic Publishers, 1990.

[16] G. Pottie and W. Kaiser. Wireless Sensor Networks. *Communications of the ACM*, 43(5):51–58, May 2000.

[17] S. Sen. Minimal cost set covering using probabilistic methods. In *Proceedings 1993 ACM/SIGAPP Symposium on Applied Computing*, pages 157–164, 1993.

[18] H. Takahashi and A. Matsuyama. An approximate solution for the steiner problem in graphs. *Math. Japonica*, 24(6):573–577, 1980.

[19] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[20] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988.

[21] Bernard M. Waxman. Performance evaluation of multipoint routing algorithms. In *Proceedings of the IEEE Infocom*, San Francisco, California, March 1993.

[22] D.J. Wetherall and D.L. Tennenhouse. The active ip option. In *Proceedings of the ACM SIGOPS European Workshop*, July 1995.

[23] P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987.

[24] Yonggang Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual Engery Scans for Monitoring Wireless Sensor Networks. Technical Report 01-745, University of Southern California, 2001.