Sweeps Over Wireless Sensor Networks

Primoz Skraba and Qing Fang
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
primoz,jqfang@stanford.edu

An Nguyen and Leonidas Guibas
Department of Computer Science
Stanford University
Stanford, CA 94305
anguyen,guibas@stanford.edu

ABSTRACT

We present a robust approach to data collection, aggregation, and dissemination problems in sensor networks. Our method is based on the idea of a *sweep* over the network: a wavefront that traverses the network, passes over each node exactly once, and performs the desired operation(s). We do not require global information about the sensor field such as node locations. Instead, in a preprocessing phase, we compute a potential function over the network whose gradients guide the sweep process. The sweep itself operates asynchronously, using only local operations to advance the wavefront. The gradient information provides a local ordering of the nodes that helps reduce the number of MAC-layer collisions as the wavefront advances, while also globally shaping the wavefront so as to conform to the sensor field layout. The approach is robust to both link volatility and node failures that may be present in real network conditions. The potential is computed by a stable diffusion process in which each node repeatedly set its potential to the average of the potentials of its neighbors. Aggregation paths are decided on-line as the sweep proceeds and no fixed tree structure is needed over the course of the computation. We present simulation results illustrating the correctness of the algorithm and comparing the performance of the sweep to aggregation trees under various network conditions.

Categories and Subject Descriptors:C.2.1[Network Architecture and Design]: Wireless communication; Network communications; Network topology

General Terms: Algorithms, Design

Keywords: Distributed Protocols, Topology, Sensor Networks

1. INTRODUCTION

Many applications of sensor networks require that data be collected from or disseminated to all nodes of the network or large subsets thereof. For example, sensor readings may need to be aggregated (in- or out-of-network) in order to compute certain statistics over the whole sensor field, or new sensor settings or code images may need to be distributed to the sensor nodes before a new round of sensing begins. These common *data collection* and *data dissemination* operations are widely used basic networking primitives that deserve careful implementation.

We investigate a novel class of methods for implementing data collection and dissemination operations using a communication pattern we call a *sweep* of the network, see Figure 1. Imagine a wavefront that traverses the physical space in which the sensor network nodes are embedded. Using such wavefronts to schedule and perform computations is very common in several disciplines dealing with modeling the physical world, such as solving partial differential equations and computing various geometric structures. In our setting, the sweep provides an orderly way to traverse the network while guaranteeing that each node is visited exactly once.

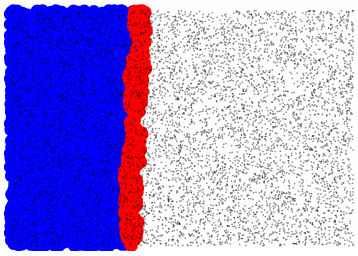


Figure 1: Nodes in the center form a wavefront that separates the swept nodes on the left and the unswept nodes on the right.

The network sweep is implemented by a narrow band of active nodes that 'moves' over the network by issuing invitations to new nodes to join the band, and dropping nodes that have already been processed and serve no other essential purpose. At any time the network is divided into three regions: the already swept and currently inactive part consisting of nodes over which the desired operation has already been performed, the active sweep band consisting of nodes that hold the currently accumulated data and implement the sweep algorithm, and the remaining unswept part.

By using simple local diffusion algorithms, we can precompute a certain *potential* at each node. The gradients associated with this potential can help guide the sweep and guarantee the correctness and efficiency of the method. We effectively solve a simple PDE, a Laplace's equation with Dirichlet boundary conditions, over the network to define

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.

a harmonic potential function whose level sets correspond to the desired wavefronts. Our algorithm, in its simplest form, is just the familiar Gauss-Seidel iteration (each node sets its value to the average values of its neighbors). The construction guarantees that no local extrema where the sweep could get stuck are present in the field. In general, such potentials and their gradients are rather resilient to link connectivity changes in the network because their values integrate information from large areas of the network. They also smooth out local variations in node density. They are used by the sweep protocol to provide global guidance so that the wavefront propagates in an orderly way without self-collision while allowing the sweep control to remain completely local. The gradients are also used to schedule invitations for nodes to join the sweep in a manner that alleviates MAC-layer collisions. Such interactions between the network and MAC layers can be efficiently implemented using the recently proposed sensor network protocol (SP) abstraction [17].

A key feature of our structure that adds to its robustness is that even though tree structures are used for partial aggregation within the sweep active band, these trees are local and are used almost as soon as they are formed. Thus, we do not need to assume the existence of a long-term stable global tree structure in the network. Furthermore, our method does not require geographic location information of the nodes or any other global network knowledge except the potential function computed during the preprocessing phase.

2. RELATED WORK

Most common approaches for data collection and dissemination in a network are tree-based. They are usually coupled with scheduling protocols that control the data flow so as the desired operation can be performed in an orderly manner [15]. However, tree structures are not robust: any single node or link failure can disconnect the tree. Obvious approaches to this problem introduce problems of their own. Retransmissions over failed links can introduce delays, while the use of more richly connected routing structures, such as DAGs, raises issues of duplicate suppression and information over-counting [15, 22]. This can be ameliorated using duplicate-insensitive encodings and approximations [16, 3], but the design of such methods is itself a challenging problem and must be thought through anew for each type of aggregation desired. In contrast, there is no long-lasting data structure needed for a particular data collection/dissemination operation in the sweep algorithm we propose. As long as radio links remain stable as the wavefront passes the relevant nodes, data will be collected/distributed correctly.

The notion of sweep in sensor networks has appeared before. In [2], information broadcast was obtained by 'wave expansion' over a multihop radio network. A polynomial heuristic scheduling algorithm was introduced which, in theory, provides collision-free communication between neighbors with bounded time-delay, assuming global synchronization, static links and a global connectivity graph known to all nodes. Collision-free communication was achieved by spatial reuse in the broadcast process, exploiting the connectivity graph. This greedy algorithm operates in phases but is not local however: coordination is required among nodes in each 'belt' — nodes with equal shortest hop distance from the broadcast node.

More recent work introduced distributed algorithms to carry out wave-like sweeps in the context of sensor fields with very regular geometry and in which sensor locations are known. [21] partitions nodes into cells and treats each cell as a 'supernode' for the scheduling algorithm; the algorithm activates spatially well separated 'edges' between supernodes. Effectively, the separation of the activated edges is ensured by their Euclidean distance, exploiting the unit disk communication graph assumed. [19] deals with sensing coverage, where a curve segment ending on two opposite borders of the sensor field periodically scans the field. A set of sensors that wake up to cover the curve form the 'hot region', whose envelope in the direction of the curve motion forms the wavefront. Each node in the hot region advances the wave by waking up all nodes whose sensing ranges (disks) intersect with its current wavefront. The difficulties facing these methods are twofold: first, accurate sensor locations are assumed but can be expensive to obtain; second, it is unclear how they would perform in the type of non-idealized connectivity observed in the real world [11].

Besides the tree-based approach discussed above, there is another line of study on data dissemination inspired by epidemic and gossiping algorithms [1, 4] developed for the Internet. *Trickle* and *Deluge* [14, 10] use a 'polite gossip' policy to adjust local 'communication rates' to local network density, meanwhile continually maintaining data consistency in a manner similar to epidemic algorithms. Such solutions have the advantage of being completely autonomous and resilient to network dynamics. However, they are only appropriate for data dissemination which is an easier problem than data aggregation as data duplication issues do not arise.

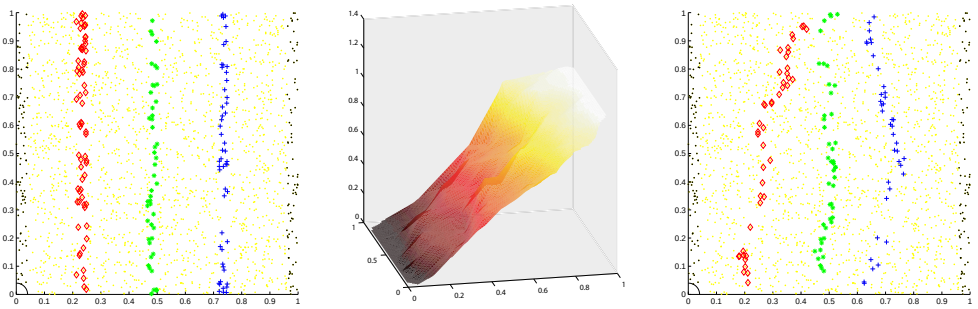
3. POTENTIAL FUNCTION AND SWEEP

Suppose that we would like to sweep over a rectangular sensor field as shown in Figure 2(i). If each sensor knows its exact location, one solution would be to have the network sweep emulate a geometric vertical sweep line moving over the field from one side to the opposite side.

However, when geographic location information is unavailable, the notion of direction given by the common coordinate system shared by all the nodes no longer exists. To compensate for this, we introduce the notion of a *potential function* over the network, whose guidance allows the sweep front to evolve in a smooth and simple manner so that it will pass over each node precisely once. We want the function to have the following properties:

1. It must be free of local extrema so that the sweep does not get stuck.
2. It must have smooth contours so that the sweep fronts are well-shaped, providing us a means for alleviating radio interference while advancing the sweep front.
3. It is desirable that the global maxima (minima) form a small connected set so that the network sweep initiation and termination can be done efficiently.

The sweep over the sensor field progresses following the gradient of this potential function from high to low values (or vice versa). The potential function locally provides a sense of direction using which the sweep could evolve in a globally consistent way. In addition, the potential function gives a local ordering among the 1-hop neighbors of any node. This



(i) (ii) (iii)

Figure 2: (i) Level sets $x = 0.0, .25, .5, .75,$ and 1.0 based on geometric coordinates of the nodes, (ii) Potential function ϕ obtained from the averaging process (iii) Level sets $\phi = 0.0, .25, .5, .75,$ and 1.0 of that potential function.

ordering can be exploited to reduce packet collisions at the MAC level.

We compute the potential function once in a preprocessing phase and reuse it for many data dissemination and aggregation operations. As the data dissemination operation is essentially a subproblem of the data aggregation operation, we focus our discussion on data aggregation from this point on.

3.1 The Model

The network is represented as a undirected graph in which each edge represents a communication link between two nodes. The links are assumed to be stable in the short term. The effect of link dynamics are discussed in section 3.6 and simulation results are presented in section 5. To define the potential function, a small subset of nodes are selected as *sink nodes* and their potential values are fixed to 0. Another small subset of nodes are selected as *source nodes*, and their potential values are fixed to 1. All other nodes are called *regular nodes*. Sources are connected to sinks via regular nodes.

3.2 Construction

The potential function can be computed during the preprocessing phase as follows. Regular nodes initially have zero potential value, and each repeatedly sets its potential value to the average of the potential values of its neighbors in the network. Formally, regular node i performs the assignment:

$$\phi(i) \leftarrow \frac{1}{|N(i)|} \sum_{j \in N(i)} \phi(j)\,, \tag{1}$$

where ϕ is the potential function and $N(i)$ is the set of neighbors of node i.

The averaging process for computing the potential on the sensor field is essentially the Gauss-Seidel iteration for solving sparse linear systems. A similar method was used in the virtual coordinate computation in Rao et. al. [18].

It is easy to see that the potential value at each regular node can only increase in the process, and that the value is bounded from above by one. Thus,

LEMMA 1. *The averaging process to compute the potential function converges.*

We observe that nodes that are disconnected from both the source and the sink nodes have their values unchanged in the averaging process. The potential values at the remaining regular nodes satisfy a system of linear equations involving their own values and the values at the source and sink nodes. This linear system is always invertible; see Proposition 3.3 in [7]. The diffusion value at each node is thus a linear combination of the values at the source and sink nodes, where the combination depends only on the network connectivity.

Using the Gauss-Seidel iteration to solve elliptic partial differential equations discretized on N grid points has been well studied. Given Dirichlet boundary conditions, the asymptotic convergence rate in 2-D is $O(N)$ [23]. In our setting, the communication graph does not form a grid mesh, and each node does not have a constant degree. Nevertheless, we observe experimentally that the convergence rate is still linear in the number of nodes in the network, see Section 5.1 for more details.

3.3 Properties

We note that in the limit, when the sensor field is a continuous domain, the resulting potential function is the solution to the Laplace's equation $\bigtriangledown^2\phi = 0$ with Dirichlet boundary conditions on the domain. The potential function is thus a harmonic function and has no local minima nor maxima. In a discrete sensor field, that is true only with certain caveats. As the potential value at each regular node is the average of the values of its neighbor, we have that

LEMMA 2. *The potential function does not have a strict minimum or maximum at a regular node.*

If a regular node is a local minimum or maximum, it must have the same value as all of its neighbors — otherwise it cannot be the average of its neighbors. The local minima (maxima) in the sensor field appear in clusters that we call *plateaus*. While there are many ways plateaus may arise in theory, in the absence of network symmetry as is often the case in sensor network, we observe that a plateau is often a group of nodes that are almost disconnected from the rest of the network. A plateau may connect to the remainder of the network through a single node called the *plateau boundary*, in which case the potential of the nodes on the plateau is the same as the potential of that plateau boundary node. A plateau may also connect to the remaining of the network via a set of nearby plateau boundary nodes that are guaranteed

to have the same potential by sharing an identical set of non-plateau neighbors. Once the potential at each node has been computed, nodes can identify themselves as plateau or plateau boundary nodes by simply comparing their potential values to those of their neighbors.

Every non-plateau node must have a neighbor with a higher potential value and a neighbor with lower potential value. The nodes that are strictly above it (with respect to the potential function) are called *upstream* neighbors, and the nodes that are strictly below it are called *downstream* neighbors. Clearly these nodes are non-plateau nodes also. From a non-plateau node, we can follow its upstream nodes to reach the source as well as follow its downstream nodes to reach the sink. It follows that

LEMMA 3. *Any non-plateau node is on some strictly monotonically decreasing path (with respect to the potential function) connecting some source node to some sink node while avoiding all the plateaus.*

3.4 The Sweep Algorithm

Given such a potential function, we can use it to guide our sweep over the network from the sources to the sinks. We exploit the gradient of the potential function to control the advance of the sweep locally, so that it follows the level sets of the potential function.

Initially, all source nodes are in the sweep and are not yet aggregated. Once all the upstream nodes of a given node have been aggregated, the node invites all its downstream neighbors to join the sweep, passes down its aggregated data to one of them, then leaves the sweep. The sweep terminates when all the aggregated data reach the sink nodes.

The set of nodes that are in the sweep at any moment is called the *sweep line*. From Lemma 3, it is easy to see that a non-plateau node is guaranteed to be invited and its aggregated data is guaranteed to reach the sink nodes. Thus

LEMMA 4. *The sweep line will pass through each non-plateau node exactly once, and the sweep will aggregate all data from non-plateau nodes to the sink nodes.*

We can augment our sweep to take care of the plateau nodes. For example, when a sensor field is well-connected, plateaus do not exist or are small, and restricted flooding is sufficient for aggregating information from such regions to the plateau boundary node(s). In such cases,

THEOREM 1. *The augmented sweep algorithm will aggregate information from all nodes in the network exactly once.*

Additionally, we would like to show that the sweep maintains certain nice topological properties. The sweep line separates the set of nodes that have already been aggregated and the set of nodes that are yet to be swept. We need to ensure that the sweep does not divide two regions of unswept nodes. In particular,

LEMMA 5. *The set of aggregated nodes can never enclose any unswept non-plateau nodes.*

PROOF. Let P be an unswept non-plateau node. By Lemma 3, there is a monotone path from P to some sink node. As P is unswept, none of the nodes in this path are swept, and thus any set of nodes inclosing P must contain an unswept node. □

Lemma 5 essentially states that the sweep line does not "self-intersect" and create pockets of unswept nodes as it advances.

3.5 Selecting source and sink nodes

We envision the sweep as starting at the source nodes and proceeding towards the sinks. When any node could potentially be a query point, building a potential function repeatedly based on different nodes can be costly. Instead, we can fix the source and sink sets so that all sweeps share the same potential function. If a query node is not in the source or sink set, it sends a message to a source set to begin the sweep and collects the aggregated information from the sink set. Here we discuss the different considerations for selecting the source and sink nodes.

First, there are structural considerations. When data is collected and disseminated via base stations or some other fixed nodes in the network, it is natural to choose such special nodes as the sinks for the diffusion potential function and the nodes on the outer boundary of the sensor field as the source nodes.

More generally, the source and and sink nodes should be small sets. A small source set reduces the start up time of the sweep while a small sink set makes final collection of the aggregated data easier. Each set must also be connected to ensure that all source nodes initiate the sweep and that all the information is accessible in one place in the network at the end of the sweep.

The sets should be well-connected with the rest of network. This important for the robustness of the sweep and the construction of the potential field. If the source or sink is poorly connected, the convergence time is longer and the resulting potential field is more sensitive to failed links. This limits how small we can make the source and sink sets. Larger sets are inherently better connected to the rest of the network by having more regular nodes as neighbors.

Determining best source and sink sets is a difficult problem in general. Although we do not know how to do it optimally, the method we propose is to pick sets of nodes which are far apart in the network. The rationale behind choosing such sets is that all gradient paths, each connecting a given node in the sensor field to the source and the sink sets, have comparable lengths, and sweeping using the resulting potential function progresses evenly.

Distant sets of nodes can be discovered by flooding. To avoid the high overhead of multiple floods, we can use the following algorithm, which gives a 2-approximation of the farthest pair of nodes in the field. Starting from a node, p, find the node q that is furthest from p. Then, find the node r that is farthest from q. The nodes q and r are the chosen pair of nodes, with one as the source and the other as the sink. Note that both q and r are on the boundary of the sensor field. The nodes q and r can subsequently recruit more nearby nodes on the boundary to enlarge the source and sink sets.

A higher-level understanding of the sensor field layout may provide insight into how to select the source and sink sets. Recently, there has been work toward this global structure of the network such as the combinatorial Delaunay complex discussed in [5] or the detection of inner and outer boundaries [5, 8, 6, 12]. Such morphological understanding can enable more intelligent source and sink selection.

3.6 Link dynamics

Because the gradient is determined by a global computation, it is relatively insensitive to small fluctuations in the connectivity structure. If the communication links change more permanently over time, we can maintain the potential function by letting nodes periodically broadcast their values, and have each node recompute its value to be the average of the latest values it knows about its neighbors. The potential at each node will converge to the correct current potential, once the network connectivity becomes stable.

During a sweep, a typical node has many downstream neighbors to which it can pass its aggregated data. This is the major advantage of using the sweep compared to a tree based approach. Even in the case when the links to all downstream neighbors of a node are out, temporarily making the node a local minimum, simple local backtracking would quickly allow the sweep to avoid that local minimum. For similar reasons, while the global convergence of the potential function is desirable, it is often not completely necessary for the sweep to progress properly.

4. THE SWEEP PROTOCOL

The sweep protocol is simple and completely local, following the program outlined in Algorithm 1. A node is initially *unswept* and remains so until it is invited into the sweep. Once it is in the sweep, it listens for possible requests for data aggregation from its upstream neighbors and aggregates any data it receives.

After all its upstream neighbors have forwarded their aggregated data and left the sweep, the node invites all its downstream neighbors into the sweep and forward its data to a selected downstream neighbor, using point-to-point communication. This is important for avoiding double counting and ensuring that the aggregated data is not lost. Once the data forwarding is successful, the node leaves the sweep.

Algorithm 1 Pseudocode for Sweep

```
repeat
   WAIT
until RECEIVE invitation from upstream node
ENTER sweep
repeat
   if RECEIVE request to aggregate then
      RECEIVE data
      AGGREGATE data
   end if
until all upstream neighbors have left the sweep
INVITE all downstream nodes into the sweep
repeat
   SELECT a downstream node
   SEND request to aggregate
until RECEIVE acknowledgement
LEAVE sweep
```

All the decisions a node makes are local and based only on its 1-hop neighborhood information. As the sweep proceeds, information in the swept parts are aggregated to nodes currently in the sweep, ensuring that when the sweep terminates, the sink nodes have the aggregate information of the whole network.

The overhead information required at each node is minimal. Each node has a node ID, a potential value, and a state (swept or not). It also keeps track of the node information of its neighbors as well as the time of the last communication with each of them.

To reduce the communication cost, nodes can obtain information about their neighbors explicitly by direct communication, or implicitly by overheard packets. In particular when a node overhears a data or acknowledgement packet from upstream nodes, it can label that upstream nodes as swept and can immediately join the sweep if it has not already done so.

The sweep algorithm uses CSMA for collision avoidance at the MAC level. As we let nodes forward their aggregated data when all their upstream neighbors have been aggregated, we effectively use the potential function to locally order the aggregation of the nodes. This ordering provided by the potential function reduces the chance of multiple nodes contending for communication channels.

If the packets being transmitted are small, a simple collision avoidance scheme such as CSMA is sufficient. For large aggregated data packets, RTS/CTS is required to reserve the channel before the packets are sent.

It may happen that a node may not hear that its upstream neighbor has left the sweep. In this case there is a timer which dictates how long a node waits before requesting state information from its upstream neighbor. If this fails, then the upstream node is considered dead and the sweep moves on. However, for some information to be missed, the upstream node must have all of its downstream links fail. A complete finite state diagram for the sweep algorithm is shown in Figure 3.

The starting and ending of the sweep require special treatment. To begin a sweep, the node initiating the sweep must first notify all source nodes. One simple way to do so is to ascend the gradient of the potential function to reach some source, then do a restricted flooding to notify them all.

The stopping criterion for the sweep is local. Once all the sink nodes have received the aggregated data from their upstream neighbors, the sweep has completed. The node initiating the sweep can periodically descend the gradient to reach a sink node to find out the status of the sweep, and if the sweep has completed, initiates a restricted flooding to collect the aggregate data from all sink nodes.

5. SIMULATIONS

In this section we investigate the correctness of our sweep algorithm, its robustness to link failures, and its aggregation speed in experimental settings and compare it with a tree-based aggregation algorithm. Two series of experiments were conducted. The first series tested the correctness and robustness of the potential function and the data aggregation algorithm on randomly generated graphs with independent random link failures and were done in MATLAB. The second series of experiments are conducted using TOSSIM [13].

5.1 Computing the potential function

In this experiment, we generated random sensor fields of various sizes on a unit square. The communication radius was chosen so that the average number of neighbors at each node was roughly 7. The sources and the sinks were selected along two opposite boundary of the square. Initially, the initial potential value at a node that was not a source nor a sink node was randomly generated, then was updated

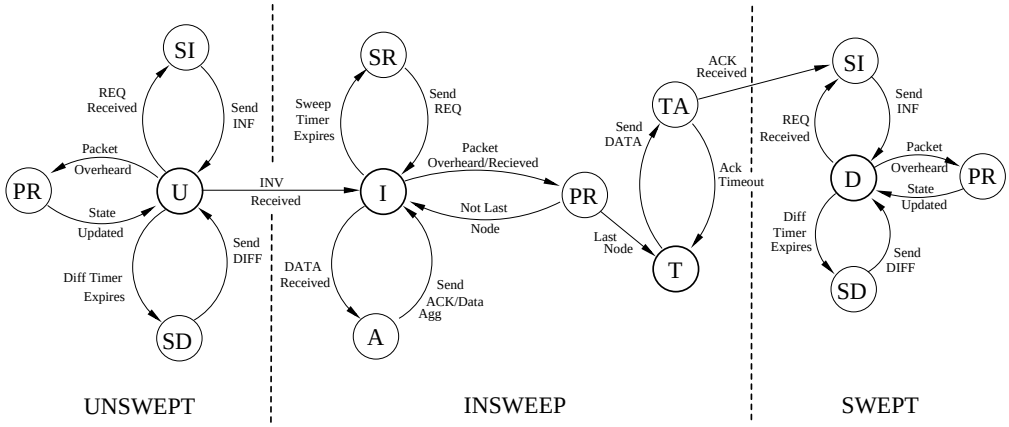


Figure 3: The state machine for the sweep. The states are: (U) Unswept; (I) InSweep; (F) Forwarding; (D) Done/Swept; (PR) Packet Received; (SI) Send State Information; (SD) Send Diffusion Packet; (A) Aggregate Data; (SR) Send Request for State Information; (TA) Waiting for Acknowledgement. The packets are: DIFF: Diffusion Value Packet; DATA: Data Packet; ACK: Acknowledgement Packet; REQ: Request for State Information Packet; INF: Information Packet; INV: Invitation Packet (A DATA or ACK packet from an upstream neighbor)

using the averaging rule. The averaging stopped when all potential values changed less than 10^{-5}.

We observed that the number of iterations it takes for the whole network to converge is roughly proportional to the number of nodes in the network, see Figure 4.

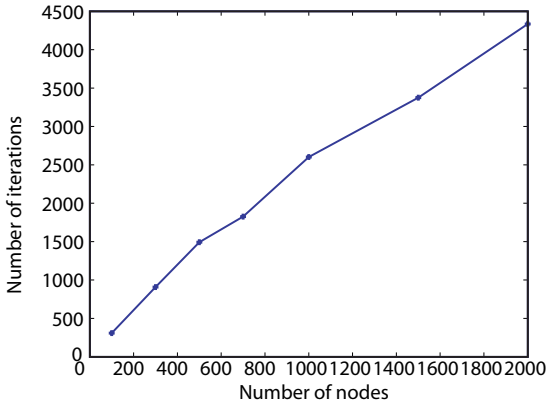


Figure 4: The number of iterations needed for convergence is proportional to the number of nodes in the network.

5.2 Flipped links and local minima

We considered a network consisting of 500 nodes randomly generated on a square. The source and the sink nodes are selected along the two opposite sides of the square. We selected the communication radius for the sensor field so that the average number of neighbors at each node was from 5 to 14. We computed a potential function on the network then randomly disabled a number of links.

Figure 5(i) shows the percentage of links that change their gradient directions after the potential function was recomputed for the new network connectivity. The figure shows that most gradient directions were stable when the network degree was 6.66 or more. For a network with degree 6.66, even when 20% of the links failed, only 8% of the gradient are now wrong.

As the links failed, some nodes became local minima, see Figure 5(ii). The number of local minima, however, was relatively small. For a network with degree 6.66, around 8% of the nodes were local minima even when 20% of the links died. We note that virtually all local minima we encountered were very shallow, and when sweeping the field, a simple local search scheme was sufficient to bypass the local minima.

As expected, when the network density increased (i.e. the number of neighbors increased), the potential function became more stable: the percentage of flipped links and the percentage of local minima became lower.

5.3 Robustness of the sweep aggregation

In this experiment we considered a simple aggregation problem of counting the nodes in the sensor field. Each node aggregated a count locally and passed down the sum when it left the sweep. We compared the performance of our sweep algorithm with a tree based aggregation approach. In the tree-based approach if a link in the aggregation tree died, we improved its performance by letting nodes pass on their aggregated sums to any nodes that were not yet aggregated. In both the sweep and tree-based approach, some of the counts accumulated could not reach the sinks, see Figure 6 for the failure rates It is clear that the sweep approach performed much better even though we did not implemented the local search algorithm to avoid the local minima. When the network degree was around 7, our sweep algorithm lost 10% of the aggregated values, while tree based algorithm lost more than 60%. The sweep algorithm was even more robust for networks with higher node degrees.

5.4 Networking performance

To verify the algorithm in more realistic conditions, the sweep was implemented on TinyOS [9] and tested in TOSSIM. The sensor network used was a 20x20 grid of 400 nodes. The source and sink sets are chosen along the opposite sides of

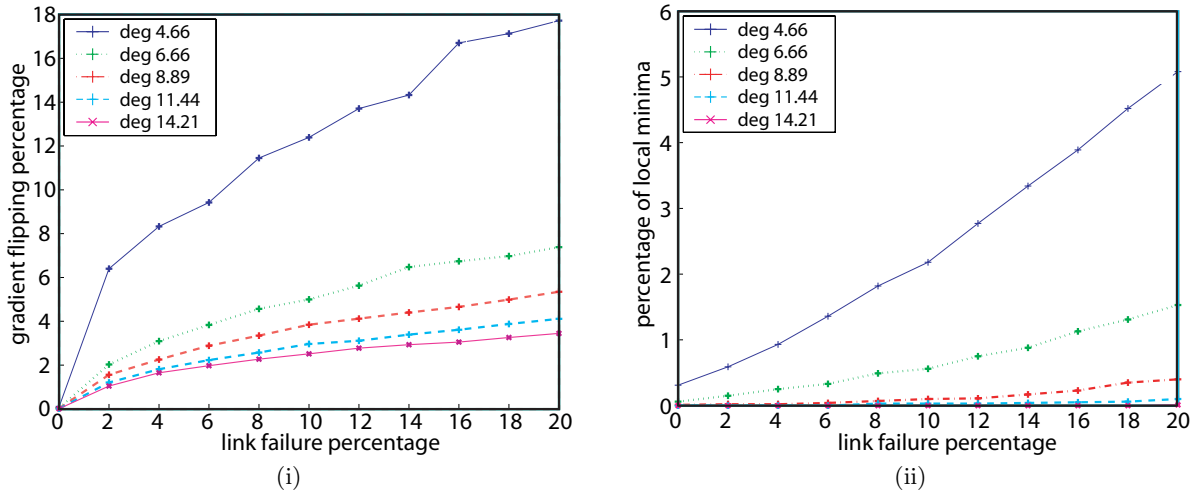


Figure 5: (i) The percentage of gradients flipped as a result of failed links for different node densities; (ii) The percentage of local minima that result from failed links before recomputation for different node densities.

Table 1: Performance results for the sweep

# of neighbors	Sweep Time(sec)	Delivery Time(sec)
4	15.8	21.8
8	48.5	52.5
12	56.8	61.9
20	100.1	101.7
8(15% loss)	62.8	68.8
8(50% loss)	151.2	159.8

Table 2: Performance of the aggregation tree

Run		Time(sec)	% of Data Delivered
4 neighbors	Epoch 1	58.5	35.8
	Epoch 2	117.5	48.3
	Epoch 3	176.2	80
8 neighbors	Epoch 1	48.2	12.5
	Epoch 2	107	59.7
	Epoch 3	161	60.7
12 neighbors	Epoch 1	24.2	11.2
	Epoch 2	66.5	42.3
	Epoch 3	109.1	61.3
	Epoch 4	151.4	68.7

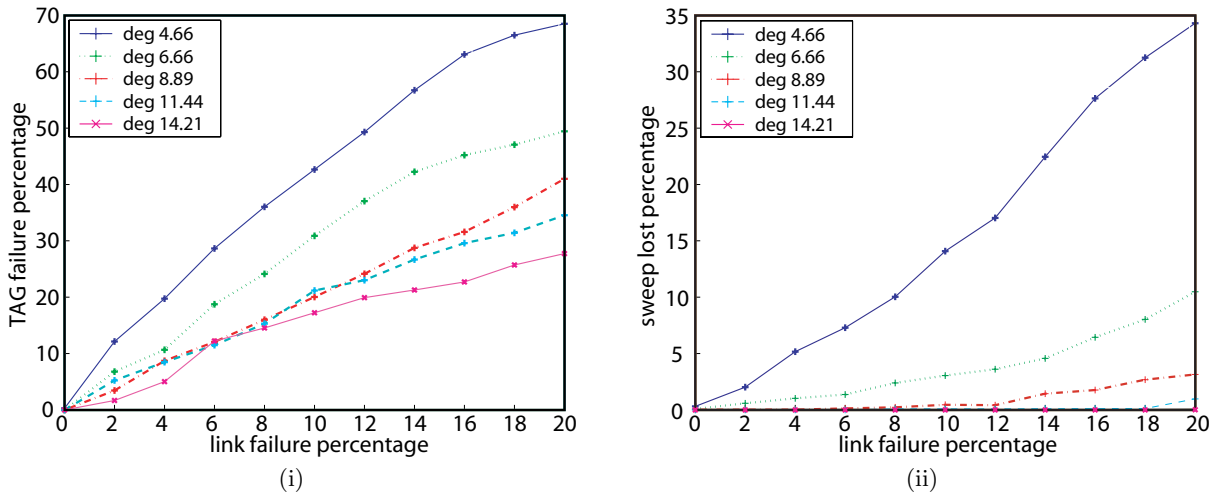
the outer boundary of the sensor field. Testing was done on networks with each node having 4, 8, 12, and 20 neighbors, with perfect links. We also considered two lossy radio models in which each node had 8 neighbors. The first lossy model had a packet loss rates of up to 15%, while the second had packet loss rates of up to 50%.

For comparison, the aggregation tree algorithm used in TAG [15] was considered. The metric considered is percent of data delivered over time. The routing tree and diffusion field were built in a preprocessing phase, and then aggregation was started. In the case of the aggregation tree, the sink boundary first flooded outward to inform all the nodes that aggregation should begin. The nodes began forwarding their data according to a schedule based on the node's depth in the aggregation tree.

We used the query nodes as the source nodes and let them initiated the sweep. When the sweep completed, the data was sent upstream to the source nodes where it was considered delivered. An example of the sweep propagation is shown in Figure 7. The node in the lower left corner began the sweep. This sweep propagated across the source boundary, then proceeded nicely to the sink boundary. The regular nodes in the middle of the field had a higher probability of collisions because they had more neighbors on average. The fact that the midsection of the wavefront progressed as fast in the middle as on the edges indicated that packet collisions did not hinder sweep progress significantly.

Figure 7: Wave fronts at roughly 10 sec intervals during the sweep showing that the sweep progressed evenly.

The complete performance results of the sweep can be seen in Table 1. The sweep time refers to the time when all nodes have forwarded their data and the delivery time refers to the time all the data is received back at the source.



(i) (ii)

Figure 6: (i) The percentage of data which is not delivered by TAG for different rates of link failure at varying node densities; (ii) The percentage of data which is not delivered by the sweep for different rates of link failure at varying node densities.

In the 4-neighbor model, the sweep proceeded very quickly across the network. With more neighbors, the sweep proceeded slower. This was because with more neighbors, fewer nodes in the sweep could transmit simultaneously, resulting in less parallelism. Introducing lossy links also increased latency due to retransmissions caused by bad links. The long duration of the sweep was caused by the current implementation which did not drop nodes from the neighbor list and forwards down the gradient, resulting on retransmissions on bad links. These implementational issues resulted in slow sweep propagation when a realistic radio model was used. We plan to implement a more optimized version of the sweep that will include link estimation and a dynamic neighbor list.

The performance of the tree implementation, which can be seen in Table 2, shows significantly worse performance than the sweep. The result was poor even though we spent a lot of effort experimenting with many combinations of parameter values for each radio model and reported only the best results. Random delays at the MAC layer caused packet losses during aggregation epochs. The scheduling problem was made harder by collisions between nodes forwarding to different parent nodes and synchronization over all levels of the tree.

6. DISCUSSION

6.1 Computing the potential function

The current computation of the potential function is rather slow — it takes $O(N)$ iterations to converge. Two possible speedup methods are worth investigation: Successive Over-relaxation (SOR) [23] methods and a multi-grid type approach.

SOR is a method of solving a linear system of equations derived by extrapolating the Gauss-Seidel method. This extrapolation takes the form of a weighted (weight ω) average between the previous iteration and the computed Gauss-Seidel iteration, successively for each component. The parameter ω is usually experimentally obtained and set so as to accelerate the rate of convergence of the iterations to the solution. Hence it could be computed given the local smoothing operator, network topology and boundary conditions.

Another way to speed up the process is to use a more elaborate initialization procedure. At the initialization stage, the source and sink nodes flood the network. Each node records its shortest hop count distances to the source and the sink, r_1 and r_2. If the potential values at the source and sink are p_1 and p_2, each node gets an initial value that equals to $r_2 \times (p_2 - p_1)/(r_1 + r_2)$. Using these initial values, the convergence for the potential computation may be faster.

6.2 Segmenting the sensor layout domain

For sensor fields with complex shapes, the gradient value might be too small across different level sets of the potential function. Such a case may then become indistinguishable from a plateau, when numerical precision is considered. One way of handling this problem is to segment the problem domain so that in each sub-domain the sensor field layout is nice, in the sense that each subnetwork is well connected. For example, in Figure 8, we can segment the domain into three sub-domains at the 'narrow bridges' region and build separate potentials in each sub-region that can then be combined to give us a nice overall potential field.

However, when we do not have prior knowledge of the network layout, we need to first decide if a segmentation is necessary and where to perform the segmentation. For a sampled shape, there exist geometric methods for segmentation based on defining a continuous flow on the samples[20] (i.e. the sensor nodes). In the case where sensor coordinates are unknown, we may employ some heuristics for identifying 'narrow bridges' of the network. A simple approach can be based on collecting shortest path statistics between a selected sample of the nodes: 1) compute the pairwise shortest paths in a flooding stage; 2) record in each node the total number of paths bypassing it; 3) increase the weight in each node for shortest path computation proportionally to the

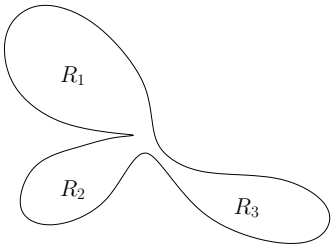


Figure 8: This illustrative example shows how an irregular region could be segmented into three subregions R_1, R_2, R_3.

number of paths it has recorded; 4) repeat 1), 2), 3) until the number of paths through each node stabilizes. The nodes with higher path counts form the 'narrow bridge'.

After the segmentation, we add a source/sink set for each well connected region and build a gradient field in the region to help with the aggregation in that region.

7. SUMMARY

In this paper, we have studied a localized, asynchronous data, energy-aware, and scalable data dissemination, collection and aggregation protocol for wireless sensor networks. Our technique uses no location information and is completely based on the link connectivity graph of the network. We take full advantage of the redundancy built in network connectivity to gain stability under link dynamics, common in the real world. There are still many unanswered questions, as discussed in section 6. However, imposing a meaningful, smooth, stable function over the communication graph and using it for various network functions can be an interesting approach for other network problems as well, including various types of routing or information brokerage, and deserves further study.

8. ACKNOWLEDGEMENT

The authors wish to acknowledge the support of DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant N00014-00-1-0637, NSF grants CCR-0204486, FRG-0354543 and CNS-0435111, and DARPA grant 32905.

The authors also wish to thank Nikola Milosavljevic (Stanford), Jie Gao (SUNY Stonybrook), and Li Zhang (HP Labs) for many useful discussions.

9. REFERENCES

[1] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 47–60, New York, NY, USA, 2002.

[2] I. Chlamtac and O. Weinstein. The wave expansion approach to broadcasting in multihop radio networks. *IEEE Transactions on Communications*, 39(3):426–433, 1991.

[3] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *In Proceedings of the International Conference on Data Engineering (ICDE04)*, 2004.

[4] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987.

[5] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2005.

[6] S. P. Fekete, A. Krller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *the First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, 2004.

[7] M. S. Floater and M. Reimers. Meshless parameterization and surface reconstruction. *Comput. Aided Geom. Des.*, 18(2):77–92, 2001.

[8] S. Funke. Topological hole detection in wireless sensor networks and its applications. In *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 44–53, New York, NY, USA, 2005. ACM Press.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, New York, NY, USA, 2000. ACM Press.

[10] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 81–94, 2004.

[11] D. Kotz, C. Newport, and C. Elliott. The mistaken axioms of wireless-network research. Technical report, Dartmouth College Computer Science Technical Report TR2003-467, 2003.

[12] A. Krller, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 06)*, 2006. To appear.

[13] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM Press.

[14] P. Levis, N. Patel, D. E. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *NSDI*, pages 15–28, 2004.

[15] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *the fifth symposium on operating systems design and implementation (OSDI)*, Boston, MA, USA, December 2002.

[16] S. Nath, P. B. Guibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *the 2nd international conference on Embedded networked sensor systems (SenSys)*, pages 250 – 262, Baltimore, MD, USA, 2004.

[17] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A unifying link abstraction for wireless sensor networks. In *The 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[18] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108, 2003.

[19] S. Ren, Q. Li, H. Wang, and X. Zhang. Design and analysis of wave sensing scheduling protocols for object-tracking applications. In *DCOSS—Distributed Computing of Sensor Systems*, pages 228–243, 2005.

[20] J. G. T.K. Dey and S. Goswami. Shape segmentation and matching with flow discretization. In *the 8th International Workshop on Algorithms and Data Structures (WADS)*, pages 25–36, 2003.

[21] N. Trigoni, Y. Yao, A. Demers, J. Gehrke, and R. Rajaraman. Wavescheduling: energy-efficient data dissemination for sensor networks. In *DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks*, pages 48–57, New York, NY, USA, 2004. ACM Press.

[22] R. G. Yonggang Jerry Zhao and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *The First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 03)*, Anchorage, AK, USA, May 11 2003.

[23] D. Young. Interative methods for solving partial differential equations of elliptic type, 1950. Docotral Thesis, Harvard University.