Data Dissemination with Geometric Structure

Jonathan Hui, Alan Newberger, Gilman Tolle

***Abstract*— Our geometry-independent bulk data dissemination protocol, Deluge, has been found to suffer from slower propagation in the central areas of a wireless sensor network due to hidden-terminal collisions caused by increasing neighborhood size, and consequent effects on timing and sender suppression. Using simulation, we investigate whether modifications to the Deluge algorithm that use explicit geometric information to manage the propagation pattern can lessen contention and reduce total time to completion. We find that the extra complexity of sharply-defined geometric patterns outweighs the marginal and network-dependent benefits that can be gained from their use, but that a simpler class of techniques based on reducing the set of possible senders according to link quality can improve dissemination performance.**

I. Introduction

Wireless sensor networks (WSNs) represent a new class of computing with large numbers of resource-constrained computing nodes cooperating on essentially a single application. WSNs must often operate for extended periods of time unattended, where evolving analysis and environments can change application requirements, creating the need alter the network's behavior by introducing new code. Unlike the traditional method of programming a node over a dedicated link, the embedded nature of these systems requires a mechanism to propagate new code over the network. However, developers face a more immediate problem. As WSN research matures, the scale of testbeds and deployments continues to grow. Testbeds sized at tens of thousands of nodes are now on the horizon, making code propagation over the network a necessity in the debugging and testing cycle. These factors suggest that *network programming* (the programming of nodes by propagating code over the network) is required for the success of WSNs. Specifically, we consider the propagation of complete binary images.

In general, the dissemination process should require a minimal amount of time. This not only reduces any service interruptions to an deployed application, but is also useful in shortening the debugging and testing cycle. However, the most current version of Deluge can be affected by density. Specifically, in dense networks, the contention and collisions caused by the random interactions of neighboring nodes harms the performance of Deluge since its suppression techniques rely on overhearing.

In this paper, we investigate and evaluate a number of geometry-based approaches, where nodes make use of information about the location of neighboring nodes to limit their interactions to a small subset of neighbors. The current version of Deluge allows interactions between any pair of nodes which can hear each other, even if they are intermittently connected.

The rest of this paper is organized as follows: Section II of this paper reviews related work; Section III presents the current version of Deluge and how its propagation dynamics are affected by spatial node density; Section IV presents the various geometry-based approaches we investigate; Section V evaluates these approaches; and concludes with Section VI.

II. Related Work

The problem we address is an important special case of reliable data dissemination. The main differences include the dissemination of data over lossy links, a constrained storage hierarchy, and the need to retain the most recent copy in order to propagate data to additional nodes as they become connected over time. Deluge builds on two bodies of prior work. The first is controlled flooding for communication in wireless and multicast networks. *Scalable Reliable Multicast (SRM)* is a reliable multicast mechanism built for wired networks [4], using communication suppression techniques to minimize network congestion and request implosion at the server.

For data dissemination in wireless networks, naive retransmission of broadcasts can lead to the *broadcast storm problem*, where redundancy, contention, and collisions impair performance and reliability [8]. The authors discuss the need to have a controlled retransmission scheme and propose several schemes, such as probabilistic and location-based methods. The experiments conducted by Ganesan et al. identify several interesting effects at the link-layer, notably highly irregular packet reception contours, the likeliness of asymmetric links, and the complex propagation dynamics of simple protocols [2].

Demers et al. propose an epidemic algorithm based on strictly local interactions for managing replicated databases which is robust to unpredictable communication failures [1]. *SPIN-RL* is an epidemic algorithm designed for broadcast networks that makes use of a three-phase (advertisement-request-data) handshaking protocol between nodes to disseminate data [5]. The epidemic property is important since WSNs experience high loss rates, asymmetric connectivity, and transient links due to node failures and repopulation. However, their results show control message redundancy at over 95% as SPIN-RL considers the suppression of only redundant request messages, and for lossy network models SPIN-RL does not perform as well as naive flooding. *Trickle* builds upon this approach by proposing SRM-like suppression mechanisms to minimize redundant transmission of control messages and pseudo-periodic advertisements to increase reliability, allow for quick propagation, and consume few resources in the steady state [7]. However, Trickle only provides a mechanism for determining when nodes should propagate code. Deluge builds directly off Trickle, by adding support for the actual dissemination of large data objects with a three-phase protocol similar to SPIN-RL.

Because reliability is of top priority, Deluge also borrows ideas from prior work in reliable data transfer protocols. *Pump Slowly, Fetch Quickly (PSFQ)* [13] and *Reliable Multi-Segment Transport (RMST)* [10] are selective NACK-based reliable transport protocols designed for WSNs where low bandwidth and high loss rates are common. Because the cost of end-to-end repair is exponential with the path length, both protocols emphasize hop-by-hop error recovery where loss detection and recovery is limited to a small number of hops (ideally one). Like PSFQ and RMST, Deluge uses a selective NACK-based approach and error recovery is limited to a single hop. However, these approaches do not consider methods for adapting to spatial node density.

Research activity directed at network programming for WSNs has been limited. Recently, TinyOS [12] has included limited support for network programming via *XNP* [3]. However, XNP only provides a single-hop solution, requiring all nodes to be within bidirectional communication range of the source. Additionally, repairs are done on a whole file basis, requiring expensive scans through external flash to discover missing data.

A more comprehensive approach to network programming is presented with *Multihop Over-the-Air Programming (MOAP)*, supporting distribution of a program image over a multihop network [11]. Deluge shares many ideas with MOAP, including the use of NACKs, unicast requests, broadcast data transmission, and windowing to manage metadata required to keep track of which segments are required. However, MOAP ignores many key design options. For example, MOAP does not fragment the image, requiring nodes to receive the entire code image before making advertisements. Thus, it does not allow the use of spatial multiplexing to leverage the full capabilities of the network. Additionally, methods to deal with the adverse effects of asymmetric links are not considered.

A *difference-based* approach to programming has also been proposed [9]. This code distribution scheme attempts to save energy by sending only the changes to the currently running code. Using optimizations such address shifts, padding, and address patching, code update information can be minimized. While a method for efficiently distributing code update information is not discussed, such difference-based methods are orthogonal and complement data dissemination protocols.

III. Deluge

The Deluge algorithm acts as a “pull” system. Nodes periodically advertise the state of their own image. If a node receives an advertisement from a neighbor with a newer or more complete image, the node requests data and continues requesting until the reliable delivery unit has been successfully received. The node then advertises the new data, and the process repeats.

A. Data Representation

In order to manage a data object that is much larger than RAM, Deluge divides the object into smaller pieces. The data object is first divided into *pages*, with fixed size S_{page}. The size of a page is chosen to fit comfortably into a RAM buffer. These pages are further subdivided into P packets of size S_{pkt}. The size of a data packet is chosen to fit into a single network message. Dividing the page into packets allows a bit-vector to represent the reception state of a page.

We divide the image into pages for several reasons. With an algorithm that requires pages to be received in order, a single node's progress can be represented by a single page number: the highest-numbered page it has received. Also, the complete reception of a page provides a natural inflection point at which a node can cease acting as a receiver and start acting as a sender, in order to support *spatial multiplexing* within a multi-hop network. Finally, pages can be given individual version numbers. This enables nodes to upgrade from previously disseminated images without requesting pages containing data that has not changed. The whole image is also given a version number, to allow nodes to determine when new data must be downloaded.

B. Dissemination Algorithm

A node executing the Deluge algorithm can be in one of three different states: *Maintain*, *Request*, or *Transmit*.

1) Maintain: The *Maintain* state represents the quiet periods when dissemination is not actively occurring. Time is divided into rounds, with round i beginning at time t_i with length $\tau_{m,i}$. At the beginning of each round, each node selects a random time r_i within the range $[\frac{\tau_{m,i}}{2}, \tau_{m,i}]$. The duration of a round ranges between τ_l and τ_h. The summary of data available on a node is represented by ϕ.

At time $t_i + r_i$, a node broadcasts a summary ϕ of the information it contains if it has not already received an identical summary ϕ' during the round. This suppression technique, as originally proposed in [7], limits the number of advertisements in a neighborhood to $O(\log n)$ where n is the number of nodes in that neighborhood. At time $t_i + \tau_{m,i}$, the node begins a new round. If no inconsistent summaries have been received, it doubles the length of the round, up to the maximum round length τ_h. If an older or newer summary has been received, it sets τ to the minimum length τ_l. This changing round length ensures that fewer advertisements are sent when the network is quiescent, but also ensures that when new data is introduced, other nodes will quickly become aware of it.

At the end of a round in which a newer data summary ϕ' was received, a node enters the *Request* state unless a request packet was overheard during the previous 2 rounds, or a data packet was overheard during the previous round. This heuristic is intended to lessen contention by preventing a node from requesting a page while other nodes in its neighborhood are receiving or transmitting data. Conversely, at the end of a round in which a request packet was received, a node enters the *Transmit* state. Request packets will be discussed in the following section.

2) Request: Once a node has become aware of a neighbor with newer data, that node enters the *Request* state and becomes responsible for reliably acquiring the next needed page. This is accomplished by sending request messages containing the number of the needed page and also containing a bit-vector representing the packets that are needed from the requested page.

After a short randomly chosen delay τ_r, a node transmits a request message to that neighbor. The random delay creates space that lowers the chance of collisions between messages from multiple requesting nodes. If no responses are heard within ω packet times, the request is repeated after waiting for a newly chosen τ_r. For a request of p packets, if ϵp, $\epsilon \in [0, 1]$ of those packets are received, the request process is allowed to continue. Once λ requests have been sent, or the page has been received, the node returns to the *Maintain* state. The λ request limit prevents a node with an asymmetrically good outbound link and poor inbound link from forcing too many redundant transmissions of data. In addition, a node can receive packets for the current page without necessarily being in the *Request* state, to take advantage of broadcast overhearing. Once each packet in the currently needed page has been received, the page is committed to persistent storage and the "needed page" counter is incremented.

3) Transmit: A node in the transmit state responds to requests by transmitting the requested packets one-by-one. For each new request message, the node takes the union of packets waiting to be sent and the packets requested by the new message. The node then continues to send the packets in circular order for fairness. Once no further packets are waiting to be sent, the node returns to the *Maintain* state.

C. Simulation Results

In this section, we discuss the propagation behavior of Deluge for a square topology. Figures 1(a) and 1(b) show the propagation of five pages from a corner node through a 20×20 network with nodes spaced 15 and 10 feet apart. These results are from a single experiment, but are representative of all other experiments performed on similar topologies. With a topology containing nodes spaced 15 feet apart, the propagation behaves as expected: the data moves at a roughly constant rate in a nice wavefront pattern from corner to corner. The irregularities are due to the non-uniform loss rates and contention.

An interesting behavior emerges as we increase density. To show this behavior, we repeat the experiment with a 20×20 network with adjacent nodes spaced 10 feet apart, increasing the density by $2\frac{1}{4}$ times. As shown in Figure 1(b), the propagation begins as it did in the sparse case. But soon after, the propagation along the diagonal begins to slow significantly while quick propagation along the edge continues and completely wraps around the edge before filling in the middle.

The root cause of this behavior is the *hidden terminal problem*, which occurs when two nodes A and C communicating within the range of node B are unable to coordinate their transmissions, thus causing collisions when transmitting to B. Nodes in the center of the network have more neighboring nodes and experience a greater probability of collisions than those on the edge of the network.

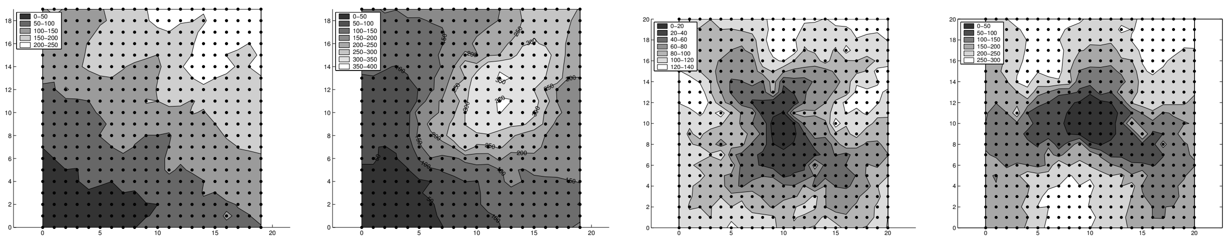
Deluge's reaction to collisions is compounded by a more subtle problem. Recall that Deluge achieves its density-aware property by taking advantage of the broadcast medium to overhear similar packets and suppresses redundancies by employing a linear backoff scheme. Thus, it relies on overhearing packets to estimate node density. When the channel is pushed to saturation, the high number of collisions can cause such a mechanism to severely underestimate the number of neighbors, stimulating transmission of more redundant messages, causing more collisions, and leading to congestion collapse. While Deluge borrows the suppression mechanisms from Trickle, it differs because it operates near channel capacity to quickly disseminate large amounts of data. The authors of Trickle studied its performance with a relatively low data rate.

In each of the setups thus far, the propagation began at a corner node. We now examine the behavior when the source node is placed at the center of the network. One might suggest that starting the propagation in the center might help to eliminate the behavior of following the edge and also decrease the propagation time by about half. We repeat the simulations with τ_r at 0.5 seconds except with a 21×21 grid and the source node at the center of the network rather than at the corner.

Figure 1(c) shows the propagation behavior for the sparse case (spacing 15). The time to reach all nodes is reduced by approximately 40%. Additionally, the propagation still does not behave in a nice circular manner. One cause is Deluge's depth-first tendency, where propagation of a single page along good links is not blocked by delays caused by poor links. Notice that the propagation speeds up as it approaches the edge of the network. Figure 1(c) shows the propagation behavior for the dense case (spacing 10). The time to reach all nodes is reduced by approximately 25%, which is significantly less than the expected 50%. Even though placing the source at the center effectively reduces the network diameter by half, Deluge is unable to take advantage of the quick edges since nodes in the center experience a greater number of collisions. However, once the propagation reaches an edge, it begins to speed around the edge as before.

IV. Geometric Techniques

As described above, Deluge is a *geometry-independent* dissemination system. The pattern in which the data propagates is the combination of



(a) Spacing 15, Corner (b) Spacing 10, Corner (c) Spacing 15, Center. (d) Spacing 10, Center.

Fig. 1. Simulated Propagation Time for 1 Page. *Figures (a) and (b) are from a* 20×20 *Grid Topology with the base node in the corner. Figures (c) and (d) are from a* 21×21 *Grid Topology with the base node in the center of the network.*

many local responses to instantaneous connectivity at the time of advertisement and request. The overhearing-based suppression of advertisements is intended to select a small subset of nodes to be senders; a subset that should create much less contention than allowing every node to become a sender would create, but should also provide as much coverage as is possible. However, the composition of this subset is also determined locally and instantly, and is not guaranteed to provide both maximal coverage and minimal contention.

By actively selecting senders and receivers, instead of relying on instantaneous connectivity and suppression, we may be able to lessen the amount of contention and increase the speed of the dissemination. Our experiments examine the effects of using explicit information about the geometry of the network to perform this active sender selection, and evaluate the effects of sender-set shaping on dissemination progress. Excessively reducing the number of senders will lessen the amount of beneficial parallelism and spatial multiplexing, and we must balance this with contention reduction.

A. Fixed-Parent Dissemination

Our first experimental geometric modification of the sender set assigns each node a single dissemination parent. The node only responds to advertisements from this parent, and requests data only from it. By giving each node a specific parent, we can create different geometric patterns within the network in order to build a contention-lessening sender set. The fixed parent approach does have one drawback, in that it eliminates the possibility of more flexible sender selection based on proximity. Our experiments examine the relative influence of these two factors.

1) Line Dissemination: As seen in the earlier experiments, propagation proceeds more quickly along the edge of the network than through the center. By creating artificial edges that pass through the center of the network, we hope to disseminate data to central nodes without suffering from the contention that normally hinders dissemination in this area. We create three different fixed-parent patterns: an edge that crosses the entire network, an edge that extends to the center of the network, and an edge that extends to the center before forking into three perpendicular edges.

In addition, we start the dissemination in three different ways: from the corner using diagonal lines, from the corner to the center of an edge then using "Manhattan" lines, and immediately from the center of an edge using "Manhattan" lines. We perform these tests because the difference in average link quality between diagonal and "Manhattan" edges is significant.

Because these patterns will clearly not reach every node, we use a *phase transition* technique by which the successful receipt of data by a node at the end of a line epidemically shifts the rest of the network from the fixed-parent model to the standard open-advertisement model. To further decrease contention, we insert a new rule that prevents a non-edge node from advertising while dissemination is proceeding along a neighboring edge.

2) Tree Dissemination: Previous work [14] has introduced Minimum Expected Transmissions routing to the wireless sensor network space. Briefly, MET routing builds a tree from a single source node

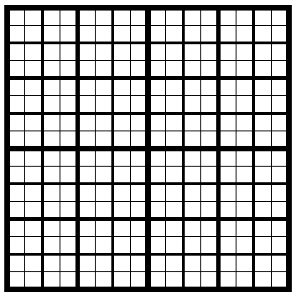


Fig. 2. The fractal dissemination algorithm on a 17x17 grid. Pages can only travel from nodes to neighbors who lie on lines equal to or 1 unit smaller in width.

to each destination node using the Bellman-Ford shortest path algorithm. Each pair of nodes within radio range is assigned an edge, and the weight of that edge is set to be the expected number of transmissions needed to successfully transfer a message between that pair of nodes. The ET value for each edge (u, v) is computed as $\frac{1}{p_{success(u,v)}} \times \frac{1}{p_{success(v,u)}}$, combining the ET values in both directions to encourage bidirectional links.

We build a MET tree according to the success probability of each link in our simulated radio network, and then set each node's dissemination parent to be its MET tree parent. We seek to determine whether restricting a node's sender to be the optimal sender for routing single messages will result in fewer retransmissions and therefore less contention as dissemination proceeds along the tree.

B. Fractal Dissemination

We also attempted to control dissemination by imposing a recursive "fractal" overlay on sensor grids. A sample overlay is shown in Figure 2. The outer edges and nodes in the horizontal and vertical center lines of the grid are assigned an index. The four quadrants created by this process are in turn quartered, and nodes on center quadrant axes are assigned indices of increasing value. During dissemination, nodes may only request pages from their neighbors where the destination node index is either equal to or one greater than the source node.

This algorithm is intended to reduce request contention and encourage propagation toward the center of the grid, and the center of each quadrant, and the center of each quadrant's quadrants, etc. There are multiple redundant propagation paths to any particular node, so faulty links in the grid should not prevent complete dissemination.

C. Density Awareness

In its original implementation, Deluge places a fixed bound on the random timers used in its suppression mechanisms. Specifically, the delay bound on the request timer is fixed to τ_r. One observation is that this random delay bound can easily become saturated in dense networks where there are essentially more nodes that wish to send packets than there are transmission slots. This, in turn, causes greater contention and collisions. Instead, we investigate whether making τ_r adaptive helps to alleviate this problem. To do so, we assume an idealized neighborhood estimation mechanism and pre-compute the number of neighbors within each node's interference range based on the simulated network topology. Nodes use this information to adjust τ_r with a deterministic function. We consider both linear and exponential functions. In the linear function, we assume that a packet transmit time is 32 ms and scale the value of τ_r as follows:

$$\tau_r = (\frac{N}{2} \cdot 32)$$

$$\tau_r = (2^{\frac{N}{16}}) \cdot 32 + 512$$

The factor of 0.5 in the linear equation represents that, on average, half of the nodes are creating hidden terminal collisions. In the exponential equation, the scaling factor of $1/16$ is intended to represent the number of packet times available in the 512ms constant term. The hope is that by increasing τ_r as the density of the neighborhood increases, less contention and fewer collisions will occur, allowing for faster propagation.

D. Neighborhood Reduction

The Deluge protocol takes a simplistic approach and does not incorporate any ability to learn about the quality of links between nodes. Thus, a node attempts to make a request from any neighboring node which advertises the needed data, and does so regardless of whether any prior requests to that node were successful or not. Instead, we consider an approach where nodes restrict themselves to requesting data only from "good" neighbors. To test the feasibility of this approach, we use two different idealized methods to select this subset of neighbors: (i) neighbors which are closest geometrically and (ii) the neighbors with the best bidirectional link quality. This approach is similar to the tree topology, but is generalized such that nodes have more flexibility in choosing which nodes to request data from. The set of "good" neighbors for each node is precomputed from the simulated network topology. We experimented with limiting the size of this subset to four and to eight nodes.

E. *On-Line Estimation*

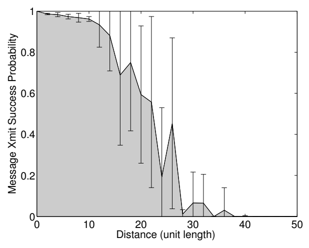
The geometric optimizations considered thus far strongly assume that each node is aware of its own Cartesian coordinates and can compute the Cartesian location of any other node based only on its address, assumptions that are unlikely to hold true in real deployments of Deluge. Therefore, we modify several of our earlier geometric techniques to obtain information from on-line estimators. We use the MET routing system from [14] in conjunction with our own geometrically aware Deluge. The MET routing algorithm uses periodic beacon messages to gather the link quality estimates needed for computation of the edge weights, as well as for propagating the cost information out from the source. We explore the effect of using these link quality estimates to perform neighborhood reduction, and of using the estimated MET tree to perform tree dissemination.

V. Evaluation

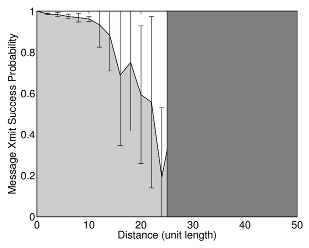
A. *Simulation Environment*

Each of the proposed optimizations to the Deluge protocol were implemented in the nesC programming language on the TinyOS platform. To evaluate and investigate the behavior of the optimizations, we used TOSSIM, a node-level simulator designed specifically for the TinyOS platform [6]. TOSSIM compiles directly from unmodified TinyOS application code and simulates communication between nodes at the bit level. One advantage of TOSSIM is that it simulates the same application logic that runs on TinyOS hardware. We focus our discussion on TOSSIM's radio model, because the radio is the most significant shared resource of a WSN and is the most important component when simulating Deluge.

Capturing sufficient detail when simulating the communication between nodes is essential since the behavior of dissemination protocols can be highly sensitive to low level factors. We have experimented with evaluating Deluge using high-level simulators, but they were unable to capture unique behaviors that appear only when we simulated a detailed model of the datalink-level network interactions and bit-level radio interactions.



(a) 50-Foot Interference Range



(b) 25-Foot Interference Range

Fig. 3. TOSSIM Packet Loss Rates vs. Distance

The network itself is modeled by a weighted directed graph $G = (V, E)$, which is static for the duration of the simulation. Each vertex $v' \in V$ represents a node and each edge $(u, v) \in E$ specifies a bit-error rate in the direction from u to v. The bit-error rate represents the probability that a given bit is flipped in transmission from u to v. Each per-link bit-error rate is a function of the distance between u and v, and is independently chosen from a random distribution derived from empirical data collected on Mica nodes. Figure 3(a) shows this model's packet success rate over distance, and the bit-error rate is derived from the packet-level success rate. Sampling the bit-error rate for each directed edge allows for asymmetric links in which the bit-error rate for (u, v) is significantly different than that for (v, u). This is important because asymmetric links are a common occurrence in WSNs.

TOSSIM treats the transmission of each individual bit as an event. If a node receives bits from multiple senders at the same time, the received bit is the union of all the transmitted bits, which will likely result in a packet that fails the CRC check executed further up in the stack. This enables the simulator to model real-world wireless interference and the effects of the hidden-terminal problem.

However, TOSSIM does have a simpler interference model as compared to real-world RF propagation. The transmission strength for each node is simulated as uniform within a 50-foot radius. This implies that while a close node may have a lower bit-error rate, it will be unable to overpower the signal of a further node. It is important to note that this aspect of TOSSIM is overly pessimistic when compared to the real world, where signal strength can fade at a polynomial rate with distance. In order to examine the sensitivity of our results to this pessimistic interference model, we also present experiments using a fixed interference range of 25 feet, as seen in Figure 3(b). The lower interference range is intended to model an environment in which poorly-connected nodes are unable to interfere with the signals of more strongly-connected nodes.

B. Performance Metrics

Without a deeper performance model of data dissemination within a homogeneous wireless network, total time to completion does not offer much insight into the differences between our geometric optimizations. Therefore, we have developed several new metrics for use alongside total completion time. These metrics are based on breaking down each simulation, and calculating the time spent using the radio and the time spent idle, for each state, as in the example shown below for the original Deluge algorithm.

```
Original Deluge
Average Time Spent (s)

Total:                 356.252

  Maintain State:        333.979
    Idle:                  313.173
    Messaging:             20.806
      Receiving:             19.474
        ADV:                   11.575
        REQ:                   4.552
        DAT:                   3.347
      Sending:               1.332
        ADV:                   1.331

  Request State:         19.810
    Idle:                  15.866
    Messaging:             3.944
      Receiving:             3.128
        ADV:                   0.548
        REQ:                   1.315
        DAT:                   1.264
      Sending:               0.817
        REQ:                   0.816

  Transmit State:        2.463
    Idle:                  1.035
    Messaging:             1.427
      Receiving:             0.264
        ADV:                   0.061
        REQ:                   0.120
        DAT:                   0.083
      Sending:               1.164
        DAT:                   1.164
```

The *Error-Free Reception Rate* represents the losses due to contention. This is calculated by counting the number of messages sent within the radio neighborhood of each node, multiplying by the average packet success rate of all the inbound radio links, and then dividing by the number of messages actually received by the node. The *REQ-DATA Ratio* is calculated by dividing the number of request packets sent by the 22-packet data page. A high REQ-DATA Ratio implies that the host was unsuccessful at obtaining needed portions of the page, and may result from the selection of a sender with a poor link, or loss of request and data messages due to contention. The percentage of time that a node, while in the Request state, spends using the radio is the *REQ Utilization* percentage. A low REQ Utilization suggests that the node is spending a large amount of time backing off, or receiving damaged messages.

Because the interference model in the simulator is quite pessimistic, we have also performed experiments using a shorter interference range. Table I presents the metrics calculated from simulation using a 50-foot interference range, and Table II contains data from experiments with a 25-foot interference range. These tables are located at the end of the paper. Figures 4(a), 4(b), and 4(c) summarize the total time performance of each experiment relative to the baseline. Note that no improvement provided more than a 25% reduction in total time, and all require a great deal of additional complexity.

C. Analysis of Results

1) Line Dissemination: We found that the speed of line dissemination has mixed results when compared to Original Deluge. Figure 5(a) presents a CDF of the single-page completion times for 440 nodes when propagating from the corner, and Figure 5(b) presents a CDF of times when propagating from the middle of the edge. The forked pattern is omitted, because we observed that the nondeterministic reception of the advertisements creating the fork tended to result in one of the three new lines completing well in advance of the others, forcing the phase transition early, and performing comparably to a single line directly across the network.

Our results demonstrate the balance between the reduced parallelism of the strict line and the excess contention of unrestricted propagation. As dissemination begins, we can see that constraining the propagation to a strict line results in a much slower rate of completion when compared to unrestricted propagation. However, once the phase transition occurs, we see a much faster rate of completion, and less of a leveling-off at the end of the dissemination. In the Diagonal Across pattern, though the propagation after phase transition was similarly fast, the extra time required to transmit the page from corner to corner over the lower-quality diagonal links resulted in an overall slower propagation. Using an artificial edge to the center of the network produced times comparable to unrestricted propagation. Our best results were obtained from sending a “Manhattan” edge across the center of the network, taking advantage of both the high-quality “Manhattan” links and the splitting of the highly-contended center region into two smaller and less-contended areas, as seen in Figure 6.

As expected, beginning the dissemination from the edge resulted in the fastest propagation, because the initial phase of moving the data from the corner to the center of the edge was rendered unnecessary.

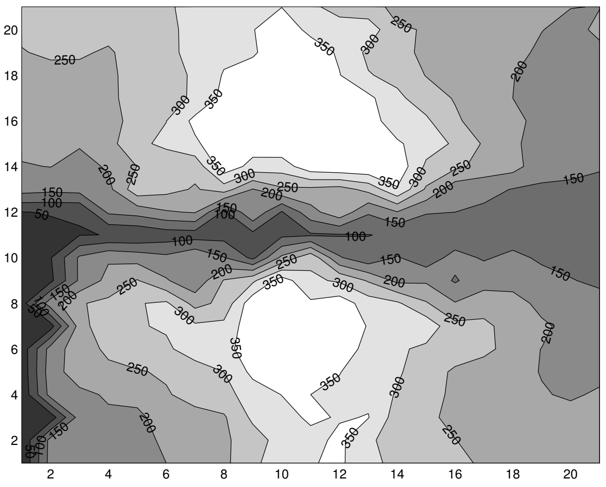


Fig. 6. Manhattan-Across Completion (s)

These results suggest that different initial positions along the edge can produce significantly different speeds.

In each of our three metrics, every line dissemination pattern performed worse than the baseline, but this is entirely produced by propagation occurring after the phase transition. Before the transition, the average reception rate is approximately 0.85, and the REQ-DATA ratio is lower than the average of any full run. We hypothesize that when the data starts from multiple internal sources simultaneously, the resulting contemporaneous sending creates an excess of contention. However, this excess contention is balanced by the smaller distance that the page must travel to reach the rest of the network.

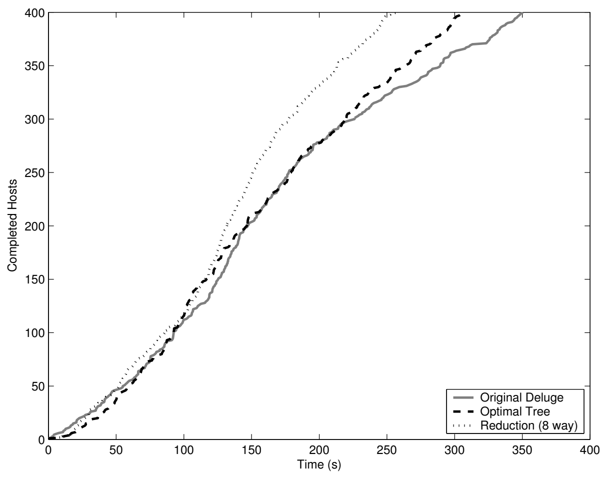
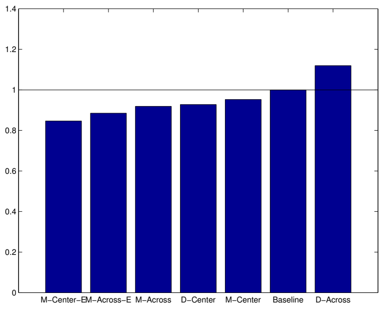
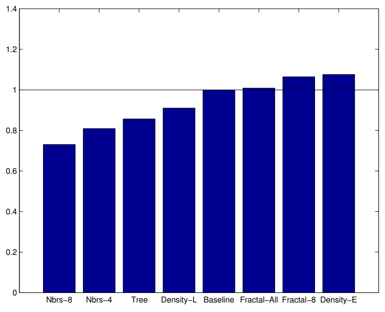


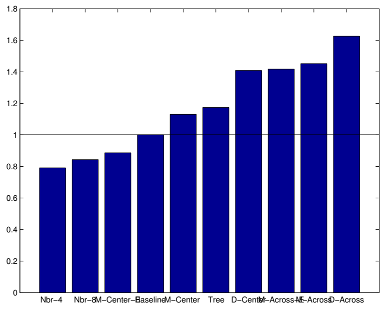
Fig. 7. Optimal Tree CDF



(a) Interference 50, 21x21

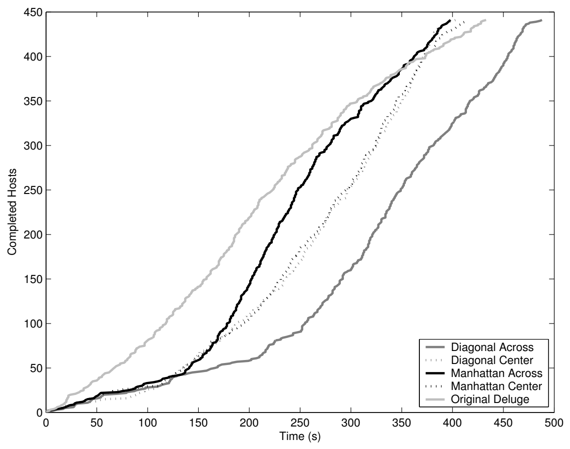


(b) Interference 50, 20x20

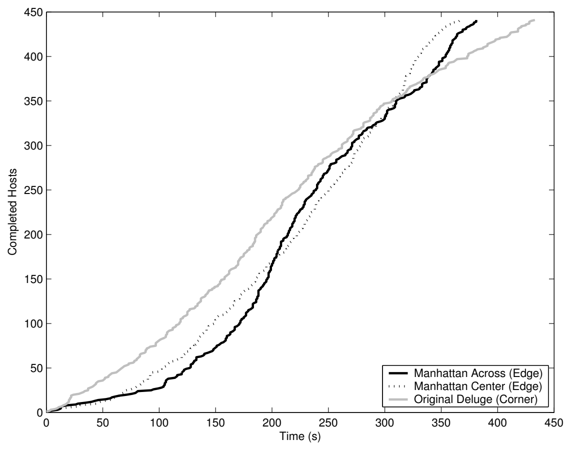


(c) Interference 25, All

Fig. 4. Total Time relative to Baseline Deluge



(a) Corner Start



(b) Edge Start

Fig. 5. Line Dissemination CDF

2) Tree Dissemination: Tree dissemination presented a propagation pattern that was very different from that of standard Deluge. In Figure 7, we see that while Deluge begins to slow down at 225 seconds, indicating that the dissemination is moving slowly through the high-contention area, tree dissemination maintains a constant speed. A node only requests data from the node to which it is most strongly connected, and contention losses are therefore not exacerbated by poor sender selection. Thus, the numbers of ineffectual requests and redundant transmissions of data are greatly reduced, and contention is reduced as a result. However, the overall finishing time is still slower than the 8-way Neighborhood Reduction, because neighborhood reduction offers equally good links, but more choice of senders. We can conclude that the fixed pattern produced by the MET method is effective, but not optimal for dissemination.

3) Fractal Dissemination: Fractal dissemination mechanisms are not superior to original Deluge. The overall propagation topology still displaying contention effects. Because of overhearing, nodes early on that are "lower" in the fractal overlay receive pages even though they did not explicitly request them. These nodes then advertise, and neighbors with the same priority respond with requests. This effect counteracts any inducement of the communication mechanism to promote page propagation first along root branches of the fractal. As shown in Figure 8, propagation of a page with the fractal overlay continues to exhibit edge effects

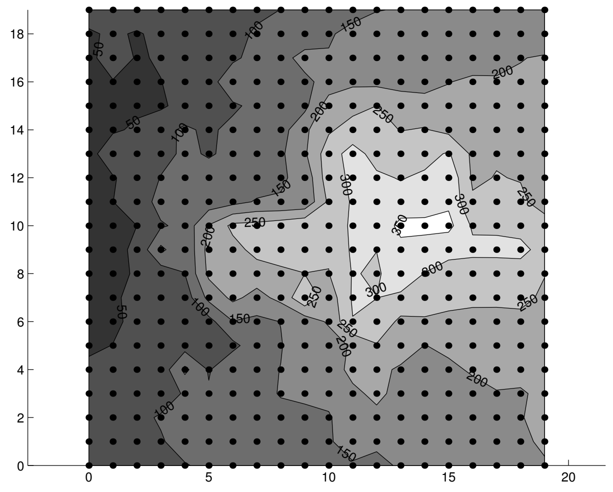


Fig. 8. Fractal with 8-way neighborhood reduction.

and center contention.

We tested the fractal overlay on a 20x20 grid with two configurations. In the first case, nodes could request pages from any other node it could hear that satisfied the fractal requirements. Nodes made less requests than original Deluge, and have a high packet reception rate. However, nodes were idle the same amount of time as original Deluge, we believe because the fractal is sabotaging any advantages due to parallelism.

In the second case, requesting nodes had to satisfy the fractal and also be a neighbor of the sender. This case is essentially sub-8-way neighbor reduction where only legal fractal neighbors are allowed. This case exhibited the best packet reception rate of all scenarios, and low request to data ratio, but request utilization was also poor here, and resulted in uncompetitive performance. Also, some nodes took a long time to complete even when all other nodes were finished. This probably occurred because of such heavy neighbor communication restrictions for nodes with poor connectivity. This behavior, as evidenced by the tail on the CDF in Figure 9, further extended the overall completion time for the scenario.

4) Density Awareness: We had mixed results with density awareness. While the linear function based on neighbors provided about a 10% improvement in total time to completion, the exponential function yielded no improvement. In general, by adjusting the request backoff bound, τ_r, based on neighborhood size, we should be able to cause less contention, and thus collisions. However, the functions used do require the use of constants, and the optimal value of these constants are unknown. One drawback of this approach is that these constants may rely heavily on the radio technology in use. To better evaluate the effectiveness of this approach, we must formulate a method for choosing these constants, which we leave for future work.

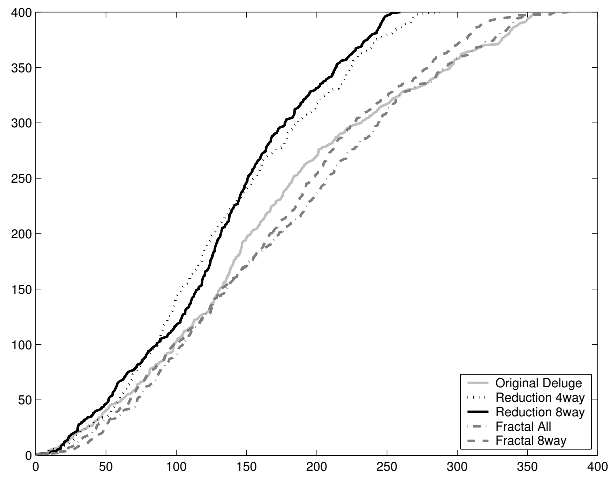


Fig. 9. Fractal and Neighborhood Reduction CDF

5) Neighborhood Reduction: Overall, the neighborhood reduction schemes provided the best performance, but the 8-way neighborhood reduction performed the best (Figure 9). There are two factors that improve its performance. First, nodes only request data from nodes which have good links, making each request highly utilized. Second, by limiting nodes to make requests to a small subset of their neighbors, we reduce the overall contention caused by request messages. It seems that the neighborhood reduction scheme provides a nice balance between restricting nodes to specific neighbors while providing flexibility in which neighbors to request data from. In contrast, the line and tree dissemination techniques provide very little flexibility.

6) On-Line Estimation: Our on-line estimation techniques were ineffective, especially in the presence of dissemination traffic. Because our estimators use periodic beacon messages with sequence numbers in order to estimate packet loss, the estimation process creates extra traffic. The estimation traffic not only caused more contention and slowed the progress of dissemination, but the dissemination traffic prevented many receptions of estimation messages. This beacon loss left most nodes unable to gather enough information about nodes to build a meaningful neighbor table, nor enough to build an MET tree.

Future work should include modification of the Deluge protocol to support passive estimation. The addition of a strictly increasing sequence number to the Deluge messages themselves should enable an estimation algorithm to calculate link qualities without introducing extra traffic. In addition, the progress of the Deluge protocol itself should offer other information sources for history-based link estimation.

D. Reduced Interference Range

Overall, we found that using a smaller interference range reduces the benefits of nearly every geographic optimization. In addition, overall time to completion drops by approximately 70%, and every contention metric is greatly improved. The only geographic technique which we believe is still helpful is the neighborhood reduction, because even a network with a reduced interference range is still susceptible to the inadvertent selection of poorly-connected sending nodes. The initiation of the dissemination from the edge instead of the corner also showed an improvement, but this effect is consistent with that seen in the extended interference range experiment.

VI. Conclusion

The use of geometric sender-set restriction techniques create a number of trade-offs, and though these techniques can result in improved dissemination performance, the existence and extent of the improvement is highly dependent on network conditions. Techniques that transmit data along a fixed path containing a small subset of nodes before allowing it to propagate epidemically are only beneficial when the network is strongly affected by hidden-terminal collisions, but excessively reduces parallelism when contention effects are less pronounced. The reduced performance and increased contention caused by initiating dissemination simultaneously from multiple interior points is mitigated by the reduction in average distance that data must propagate to reach the entire network. More complex geometries imposed over the network, like trees and recursive fractals, may smooth out the response of the network to contention, removing contention hot-spots but slowing the propagation through areas that would not normally suffer from contention.

The best results are gained, paradoxically, from the techniques that are the least “geometric”. Restricting the set of senders to those with the best links removes the possibility of a stray long-link advertisement inducing a storm of unsatisfiable requests, and should improve performance. Neighborhood reduction can be performed with active or passive estimation, even in a non-geometric network. Making backoff timers responsive to density estimates reduces the negative effects of request implosion, and this technique can also be applied with estimation and little explicit geometric information.

Additionally, our examination of the interference model used in the TOSSIM simulation environment suggests that room for future work exists in developing a more realistic model that establishes an active relationship between link quality and interference probability.

References

[1] A. Demers, D. Greene, C. Hauser, W. Irish, and J. Larson. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 1–12. ACM Press, 1987.

[2] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report UCLA/CSD-TR 02-0013, UCLA, 2002.

[3] J. Jeong, S. Kim, and A. Broad. *Network Reprogramming*. University of California at Berkeley, Berkeley, CA, USA, August 2003.

[4] S. K. Kasera, G. Hjálmtýsson, D. F. Towsley, and J. F. Kurose. Scalable reliable multicast using multiple multicast channels. *IEEE/ACM Transactions on Networking*, 8(3):294–310, 2000.

[5] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless Networks*, 8(2-3):169–185, 2002.

[6] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire tinyos applications. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*. ACM Press, November 2003.

[7] P. Levis, N. Patel, S. Shenker, and D. Culler. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. Technical report, University of California at Berkeley, 2004.

	Grid	Time(s)	Reception Rate			REQ-DATA Ratio			REQ Utilization (%)		
			Mean	Max	Min	Mean	Max	Min	Mean	Max	Min
line: across	21	496	0.58	-	0.28	1.39	6.82	0.0	17	50	0
line: center	21	411	0.55	-	0.29	1.49	7.73	0.0	19	48	0
line: m-across	21	407	0.54	-	0.29	1.35	5.73	0.0	20	52	0
line: m-center	21	422	0.52	-	0.27	1.58	7.32	0.0	19	45	0
line: me-across	21	392	0.52	-	0.25	1.46	5.27	0.0	20	45	0
line: me-center	21	375	0.51	-	0.25	1.61	8.77	0.0	20	44	0
tree	20	305	0.72	-	0.42	0.63	5.41	0.0	24	60	0
fractal: all	20	359	0.74	-	0.48	0.87	4.14	0.0	22	48	0
fractal: 8way	20	379	0.93	-	0.71	0.32	1.50	0.0	23	51	0
density: linear	20	324									
density: exp	20	383									
reduction: 4way	20	288	0.83	-	0.57	0.28	2.0	0.0	28	70	0
reduction: 8way	20	260	0.74	-	0.5	0.4	2.27	0.0	28	79	0
baseline	21	443	0.57	-	0.35	1.20	3.73	0.05	22	44	11
baseline	20	356	0.63	-	0.36	1.16	4.5	0.05	22	54	13

TABLE I

UTILIZATION METRICS: FULL INTERFERENCE RANGE

	Grid	Time(s)	Reception Rate			REQ-DATA Ratio			REQ Utilization (%)		
			Mean	Max	Min	Mean	Max	Min	Mean	Max	Min
line: across	21	187	0.89	-	0.62	0.41	8.36	0.00	-	-	-
line: center	21	162	0.87	-	0.61	0.40	1.59	0.00	36	73	0
line: m-across	21	167	0.87	-	0.64	0.39	1.05	0.00	35	70	0
line: m-center	21	130	0.84	-	0.55	0.42	1.41	0.00	39	73	0
line: me-across	21	163	0.88	-	0.61	0.38	1.09	0.00	36	68	0
line: me-center	21	102	0.82	-	0.58	0.44	1.18	0.00	38	75	0
tree	20	135	0.98	-	0.67	0.13	0.73	0.00	38	99	0
reduction: 4way	20	91	0.95	-	0.73	0.11	0.55	0.00	38	79	0
reduction: 8way	20	97	0.90	-	0.62	0.15	0.68	0.00	41	100	0
baseline	21	114	0.83	-	0.54	0.43	1.09	0.05	40	71	20
baseline	20	117	0.86	-	0.49	0.47	1.23	0.00	41	85	15

TABLE II

UTILIZATION METRICS: REDUCED INTERFERENCE RANGE

[8] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 151–162. ACM Press, 1999.

[9] N. Reijers and K. Langendoen. Efficient code distribution in wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 60–67. ACM Press, 2003.

[10] F. Stann and J. Heidemann. RMST: Reliable data transport in sensor networks. In *Proceedings of the First International Workshop on Sensor Net Protocols and Applications*, pages 102–112, Anchorage, Alaska, USA, April 2003. IEEE.

[11] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical report, UCLA, Los Angeles, CA, USA, 2003.

[12] University of California, Berkeley. Tinyos. http://www.tinyos.net/, 2004.

[13] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 1–11. ACM Press, 2002.

[14] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of multihop routing in sensor networ ks. In *Proceedings of the First ACM Conference on Embedded Networked Sen sor Systems*, Los Angeles, CA, Nov. 2003.