GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks *

Fan Ye, Gary Zhong, Songwu Lu, Lixia Zhang
Computer Science Department
University of California, Los Angeles, CA 90095-1596
Phone: 310-825-4838 Fax: 310-825-2273

{yefan, gzhong, slu, lixia}@cs.ucla.edu

ABSTRACT
Although data forwarding algorithms and protocols have been among the first set of issues explored in sensor networking, how to reliably deliver sensing data through a vast field of small, vulnerable sensors remains a research challenge. In this paper we present GRAdient Broadcast (GRAB), a new set of mechanisms and protocols which is designed specifically for robust data delivery in face of unreliable nodes and fallible wireless links. Similar to previous work [13, 14], GRAB builds and maintains a cost field, providing each sensor the direction to forward sensing data. Different from all the previous approaches, however, GRAB forwards data along a band of *interleaved mesh* from each source to the receiver. GRAB controls the width of the band by the amount of credit carried in each data message, allowing the sender to adjust the robustness of data delivery. GRAB design harnesses the advantage of large scale and relies on the collective efforts of multiple nodes to deliver data, without dependency on any individual ones. We have evaluated the GRAB performance through both analysis and extensive simulation. Our analysis shows quantitatively the advantage of interleaved mesh over multiple parallel paths. Our simulation further confirms the analysis results and shows that GRAB can successfully deliver over 90% of packets with relatively low energy cost, even under the adverse conditions of 30% node failures compounded with 15% link message losses.
keywords: sensor networks, robust data delivery

1. INTRODUCTION
Recent technology advances in low-cost, low-power chip designs have made the deployment of large-scale sensor networks economically feasible. Thousands or even millions of small, inexpensive and low-power sensors, such as Berkeley Motes[7], can be quickly deployed to monitor a vast field. The sensors collectively sense the environment and deliver sensing data via a wireless channel. In the near future such sensor networks may play an important role in various applications ranging from precision agriculture to disaster recovery and military surveillance. On the other hand, the above mentioned potential applications also present great challenges to reliable sensing data delivery. Severe operational conditions (e.g. rain, fire, strong wind, or high temperature) may easily damage individual sensors, resulting in a constantly changing topology. Wireless communications among these small, power-limited sensor nodes are also prone to errors. Furthermore, the short transmission range of small sensors means that sensing data may travel a large number of hops to reach intended destinations, with potential transmission errors and unexpected node failures at each hop.

In this paper we propose GRAdient Broadcast (GRAB) to address the problem of robust data forwarding to a data collecting unit (called the *sink*) using unreliable sensor nodes with error-prone wireless channels. The objects or events to be monitored are called *stimuli*. All the sensor nodes that detect the same stimulus collectively elect one that generates a sensing report on behalf of the group. We call such a node a data *source*. The sink builds and maintains a *cost field*. Each node keeps the cost for forwarding a packet from itself to the sink. Nodes "closer" to the sink have smaller costs. Instead of a sender appointing specific receivers to continue forwarding, in GRAB each receiver decides whether it should forward a packet by comparing its own cost to that of the sender. As a result, sensing data follow the direction of descending cost to reach the sink. Since multiple paths exist between a source and the sink, a source assigns a *credit* to each report it sends out to control the degree of path redundancy. The credit is some *extra budget* that enables a packet to be forwarded over a mesh of interleaved paths, each of which has a cost not greater than the total budget of the credit plus the cost of the source. The amount of credit determines the "width" of the mesh, thus the degree of robustness and overhead.

GRAB design harnesses the advantage of large scale. It achieves system robustness by relying on collective efforts from multiple sensors without dependency on any individual one. A packet is forwarded over multiple paths, which improves reliability. The interleaving of such paths provides tolerance of node failures or link errors along any individual path, thus significantly increasing data delivery robustness as shown by our analysis (Section 3.2). Since each receiver decides whether it should forward a packet, the sender does not need to keep state information about to which neighbor to forward data. The elimination of such explicit path state also removes the overhead and complexity in repairing paths for failed nodes or broken links; a packet simply travels through whichever working nodes to reach the sink. The credit provides a control knob to tune the robustness degree and total cost.

*This work is supported in part by the DARPA SensIT program under contract number DABT63-99-1-0010. Lu is also supported by an NSF CAREER award (ANI-0093484).

The rest of the paper is organized as follows: We present the design of GRAB in Section 2 and analyze its robustness and complexity in Section 3. In Section 4 we evaluate its performance and compare it with an existing protocol [8]. Section 5 discusses the differences between GRAB and related efforts in sensor networking. Section 6 summarizes the work and sketches out our future plan.

2. GRAB DATA FORWARDING PROTOCOL

2.1 Design Overview

In this paper we assume the following sensor network model (as shown in Figure 2): Large numbers of small, stationary sensor nodes are deployed over a field. The user collects sensing data via a stationary sink (at the bottom of the figure) that communicates with the network. Each stimulus is detected by multiple nearby sensor nodes and one of them generates reports as a source (at the upper left corner). Due to the limited radio range, reports are forwarded over many hops before reaching the sink. Nodes can tune their transmitting powers to control how far the transmission may reach. Such power adjustments save energy and reduce collisions whenever possible [1]. The nodes use CSMA MAC to communicate, which is less reliable than 802.11 DCF that has RTS/CTS/ACK exchange. External noises and disturbances may further exacerbate the channel quality. Due to the harsh environment, sensor nodes suffer unpredictable, and possibly frequent failures.

We use an example of one sink and one stimulus to illustrate how GRAB works. To collect data reports, the sink first builds a *cost field* by propagating advertisement (ADV) packets in the network. The cost at a node is the minimum energy overhead to forward a packet from this node to the sink along a path. We assume each node can estimate the cost of sending data to nearby neighbors (e.g., based on the signal-to-noise ratio of neighbors' transmissions). The costs of all nodes in the network form the cost field[2]. If we imagine each node be elevated to a height proportional to its cost, the whole cost field would look like a funnel (see Figure1):nodes "closer" to the sink have smaller costs and are "lower", while those "farther" away have greater costs and are "higher".

The cost field gives the global direction towards the sink implicitly. When a node forwards a packet, it does not designate which nodes are the next hop. It simply includes its own cost in the packet. Only neighbors with smaller costs may continue forwarding the packet. Neighbors with higher or equal costs silently drop the packet because they are at the "wrong" direction. Thus packets travel in a cost field like water flows down to the bottom of a funnel: they follow the direction of decreasing cost to reach the bottom of the cost field, which is the sink. Multiple paths of decreasing cost exist; they interleave and form a forwarding mesh.

The election of a source follows the same mechanism. We want only one node to generate the report since it would be a waste of resources if every node detecting the stimulus sends a report. The stimulus creates a field of sensing signal strength, the "shape" of which is similar to that of the cost field. Each node broadcasts a message indicating its signal strength (with some random delay to avoid collision). A node rebroadcasts its signal strength whenever it hears a neighbor's message with a weaker signal, but stops broadcasting when it hears a stronger one. This way, messages roll towards the center of the signal strength field. Finally the node with the strongest signal generates a report. We call this node the Center of Stimulus (CoS).

CoS election and data forwarding utilize the same concept of gradient field where the intensity of a property varies over distance. The differences are: The signal strength field already exists in the physical world, whereas the cost field is an artifact created by the sink; nodes farther to the stimulus have weaker signals, but nodes farther to the sink have greater costs; when a stimulus is detected, data come from all directions to the center, but for forwarding, they come only from the direction of the source.

Since usually more than enough paths of decreasing cost exist, we want to limit the "width" of the forwarding mesh. Otherwise the packet would follow every possible path of decreasing cost, creating excessive redundancy and wasting resources. To control the "width", a source assigns a credit α to the packets it sends out. The credit is some extra budget that can be consumed to forward the packet. The sum of the credit and the source' cost (i.e., $\alpha + C_{source}$) is the total budget that can be used to send a packet to the sink along a path. A packet can take any path that requires a cost less than or equal to, but not beyond the total budget.

The amount of credit controls the redundancy of the mesh flexibly. If there is no credit, the packet can only be forwarded along the single minimum cost path of the source; when more credit is added to increase the budget, more paths are available to deliver the packet. Such paths intertwine with the minimum cost path and form the forwarding mesh *dynamically* through the combined effect of the cost field and the credit value carried in each packet (see Figure 2 for an example).

A final point we would like to make before presenting the design is the number of sources and sinks the GRAB design can support. To simplify the presentation we use an example of one stationary source (CoS) and one stationary sink. However we point out that GRAB supports data forwarding from multiple, mobile *stimuli* as well. When a stimulus, such as a tank, moves to another location, a different CoS sensor is elected to generate the report; the old CoS node stops reporting automatically because it finds itself no longer at the center of the stimulus. The case of multiple stimuli works in a similar way. The exact details are not presented in this paper, which focuses on robust data forwarding.

In the rest of this section, we give a brief summary in Section 2.2 about an algorithm proposed in our previous work [16] to build the cost field efficiently. Then we present the GRAB forwarding algorithm in Section 2.3.

[1] Some existing hardware [7] already have different levels of transmitting power.

[2] The cost may take different forms such as hop number, energy overhead or even physical distance. The current energy form is meant to save the scarce energy resources of nodes. The ADV packet may also carry user interests to task sensor nodes.

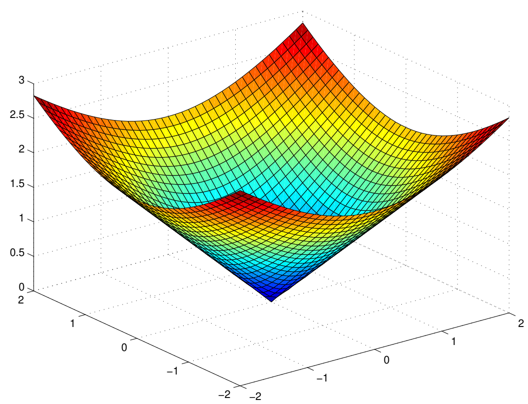


Figure 1: The "shape" of the cost field is like a funnel, with the sink sitting at the bottom. Packets follow the decreasing cost direction to reach the bottom of the cost field, which is the sink.

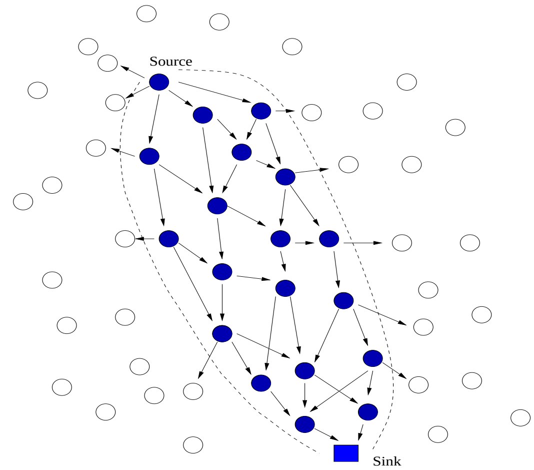


Figure 2: The forwarding mesh starts from a source and ends at the sink. The black nodes forward the packet to the sink collectively. Notice that some nodes outside of the mesh also receive the packet but do not forward it.

2.2 Building and Maintaining the Cost Field

The cost field can be built in the following straightforward way. A sink broadcasts an advertisement packet (ADV) announcing a cost of 0. Each node initially has a cost of ∞. When hearing an ADV packet containing the cost of the sender, a node calculates its cost by adding the link cost between itself and the sender to the sender's advertised cost. It compares this cost to the previously recorded one and sets the new cost as the smaller of the two. Whenever it obtains a cost smaller than the old one, it broadcasts an ADV packet containing the new cost. The "rippling" of ADV packets from the sink outwards builds the cost field for the sink[3].

The problem with the above method is excessive ADV messages, which prevent it from scaling to large numbers of nodes. Before a node settles with the minimum cost, it may hear many ADV packets, each of which results in a smaller cost than the previous one. Thus the node broadcasts many ADV packets. To build the cost field in a scalable manner, we proposed a waiting algorithm in [16] and proved that it ensures each node broadcasts only once, and with its minimum cost.

A node's cost depends on the topology. The topology changes as nodes fail, exhaust energy, or new nodes are deployed. The initially built cost field thus becomes inaccurate. Although the GRAB forwarding protocol is highly robust against inaccuracies in cost field (we will see that in Section 4), the cost field should be refreshed on time to keep the forwarding efficient.

To avoid the overhead of periodic refreshing, we choose an event-driven design. The sink keeps a profile about the recent history of data reports from the source. It includes the success ratio (packets are sequentially numbered so a sink can calculate success ratio), the average consumed budget, the average number of copies received per packet for recent reports. Once a new packet is received, the sink compares the parameters of the packet to those in the past. If a parameter differs from its past by a certain threshold, the sink broadcasts a new ADV packet to rebuild the cost field. Due to space limit, more details are in a technical report[17].

The rationale behind the event-driven refreshing is that topology changes bring variations in data delivery. By monitoring certain parameters which reflect the quality of data delivery, we can tell the amount of change that has happened. Only major changes that make the data delivery deteriorate beyond acceptable levels trigger refreshings. The forwarding algorithm itself is robust enough to withstand significant amount of changes, which will be shown in Section 4.

Before we proceed to the forwarding algorithm, we want to point out that [16] solves only the problem of building the cost field. It does not address robust data delivery with unreliable sensor nodes, which is the centerpiece of this paper.

2.3 Credit-based Robust Forwarding Mesh

After the cost field is built, a source sends a report that carries a "credit." At each hop only nodes that have costs smaller than the sender can forward the packet. To ensure a robust forwarding mesh, we need to address three issues.

First, how to expand the mesh to a sufficient width quickly starting from the source. To be robust, the mesh should be wide enough to contain sufficient parallel nodes (paths). When there are node failures or packet losses, a sufficient width ensures some nodes can still deliver packets successfully to the next hop. Since there is only one node (the source) at the first hop, we need to expand the mesh to a sufficient width quickly. Otherwise, the delivery may fail before the mesh grows wide enough.

[3]This is originally how GRAB gets its name. "Gradient" stands for the cost, the broadcast of gradients builds the cost field. Notice that although the same word is used, the "gradient" here is completely different from that in [8], where it stands for a vector pointing to a next hop neighbor.

Second, after establishing a sufficient width of the mesh, how to maintain it. Since a node may fail unexpectedly or may not receive the packet, the number of parallel nodes that forward a packet tend to decrease from one hop to the next. If no measure is taken to counteract this tendency, the mesh can narrow down later.

Finally, how to prevent packets from traveling along some devious paths or diverting too much from the direction of the sink. For any sender, roughly half of its neighbors have smaller costs. If all such directions of decreasing costs were followed, the forwarding could diffuse into a sector-shape, in which many packets divert significantly from the direction of the sink. We want to stop packets from following such diverting paths.

2.3.1 Solutions to the Issues

To address the first two issues, we divide the total amount of credit among different hops in a way where beginning hops can consume larger shares of the credit, while later hops consume some, but smaller shares. This is because the share of credit a node receives decides whether it can expand the mesh. If a node does not have any "bonus" to use but has only a budget equal to its cost, it can reach only its next hop neighbor on the minimum cost path, without expanding the mesh. When a node has some "bonus" (credit) to use, it can consume more budget, reaching more receivers, thus expanding the mesh. Beginning hops use more credit to expand the mesh quickly while later hops do not need as much credit because they do not need to expand the mesh. However, they should also receive certain credit to prevent the mesh "narrows down" due to node failures and packet losses.

Now, how to solve the last issue? If a packet has been following a quite devious path, or divert from the direction of the sink too much, it will consume much credit, without traveling proportionately close towards the sink. An analogy to this is spending most of a month's budget in a few days. Thus by comparing the remaining credit to the remaining "distance" ahead, nodes can discover and terminate such packets.

We achieve the desired division of credit among different hops and detect diverting packets through one mechanism: a threshold function indicating the relative "position" of a node between the source and the sink. Each packet carries certain dynamic state so that a node can calculate how much credit remains unconsumed. The node then compares the remaining credit to the threshold, thus discovering if the packet has already consumed too much credit, or there is still enough to use. The format of the threshold function is chosen carefully so that it divides the credit as required. We first describe what are carried in a packet, then present the detailed forwarding algorithm.

2.3.2 GRAB Forwarding Algorithm

A packet carries the following fields:

- α: the amount of credit assigned to the packet at the source. It determines the "width" of the forwarding mesh. This field does not change as the packet travels towards the sink.
- C_{source}: the cost of the source to send a packet to the sink. It is used to calculate the threshold. This field does not change at different hops, either.
- $P_{consumed}$: the amount of budget that has been consumed from the source to the current hop. It is the accumulative sum of the initial cost used by the source to broadcast the packet and the cost used to forward the packet at each hop.
- C_{sender}: the cost of the current sender that broadcasts the packet.

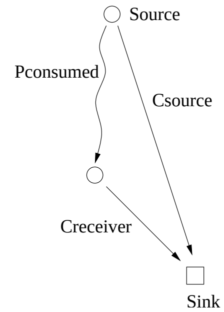


Figure 3: Calculating used credit: if a packet were to reach the sink through the receiver, at least it needs a total budget of $P_{consumed} + C_{receiver}$. This amount less C_{source} is the amount of credit that has been used

After a CoS assigns an α to a data report, it fills the above three fields and broadcasts the packet. Since only receivers with smaller costs may forward the packet at each hop, the packet is forwarded by successive nodes of decreasing costs. It cannot traverse back to the same node, thus loops will not occur.

If a receiver finds its cost $C_{receiver}$ less than C_{sender} in a report, it calculates and compares two values R_α and R_{thresh} as follows.

$$R_\alpha = \frac{\alpha - \alpha_{used}}{\alpha} \quad (1)$$

$$R_{thresh} = (\frac{C_{receiver}}{C_{source}})^2 \quad (2)$$

where

$$\alpha_{used} = P_{consumed} + C_{receiver} - C_{source} \quad (3)$$

In the above equations, α_{used} stands for the amount of credit that has been consumed, $P_{consumed} + C_{receiver}$ is the least amount of total budget required should this node forward the packet via any path to the sink successfully (See Figure 3). This minimum amount is achieved when the packet takes the minimum cost path from this node to the sink. The "extra" amount of this to C_{source}, is the credit consumed. So R_α represents the fraction of credit still available for this node and later hops. R_{thresh} indicates the normalized "distance" from this node to the sink. Both numbers range between 0 and 1.

The node then compares R_α to R_{thresh}. If R_α is greater than R_{thresh}, we consider the node has sufficient credit to use. It broadcasts at a power to reach multiple neighbors towards the sink (we define neighbors with smaller costs as this node's *nearer neighbors*). The amount of the power depends on the degree of robustness desired. A higher robustness requires more nearer neighbors. In current design, we let the node broadcast the packet at a power to reach three closest nearer neighbors. [4] The node knows this power from the ADV messages it received during cost field building [16]. It increases $P_{consumed}$ by the amount of budget that it is going to consume in broadcasting and sets C_{sender} to its cost. Then it broadcasts the packet to reach those three nearer neighbors.

If R_α is smaller, however, the node does not have sufficient credit and should forward the packet along its minimum cost path to minimize the total cost. Thus the node sends the packet to the next hop neighbor on its minimum cost path. It sets $P_{consumed}$ and C_{sender} in the packet accordingly. For completeness, the pseudo code is shown in the Appendix.

To reduce collisions, a forwarding node always waits for some random time before sending the packet, so that neighboring senders do not broadcast simultaneously and result in collisions.

It is possible that a node receives multiple copies of the same packet from different upstream nodes, and each copy has enough credit to use. To suppress such duplicates, each node maintains a cache which stores the signatures of recently forwarded packets. The signature of a packet can be the header of the packet, or a hash of the packet calculated on demand. It serves as an identifier to distinguish packets. If the signature of a received packet is found in the cache, the packet is dropped. Notice that this is an optimization technique, not a fundamental requirement of the design.

Although a number of parameters are needed in forwarding a packet, only α has impact on the delivery robustness and needs to be set appropriately. Other parameters depend solely on the topology and cannot be set arbitrarily. Generally speaking, α should be large enough to ensure robustness but not too big to cause excessive energy consumption. We will further evaluation its impact in Section 4.1.1.

3. THEORETICAL ANALYSIS

In this section, we analyze how the chosen threshold function favors beginning hops and gives them more shares of the credit to allow quick expansion of the mesh. Then we compare the robustness of interleaved forwarding mesh and multiple disjoint paths. Finally we give the complexities of the algorithm.

3.1 Credit Division among Different Hops

We derive the maximum share of credit a hop can consume under the requirement that sufficient credit remain at each hop. This is achieved when the remaining credit ratio R_α is equal to threshold R_{thresh}. To simplify the analysis, we use the following simple model: There is a chain of nodes from a CoS to the sink; the cost of each node is proportional to the number of hops to the sink. Denote node A's cost as C_A. We have

$$\frac{\alpha - (P_{consumed} + C_A - C_{source})}{\alpha} = \left(\frac{C_A}{C_{source}}\right)^2 \qquad (4)$$

$$P_{consumed} = -C_A - \alpha \cdot (\frac{C_A}{C_{source}})^2 + \alpha + C_{source} \qquad (5)$$

Taking derivatives for $P_{consumed}$, we have

$$\frac{\partial P_{consumed}}{\partial C_A} = -\left(1 + 2\alpha \frac{C_A}{C_{source}^2}\right).$$

Then, the allowed energy consumption at a hop is:

$$\Delta P_{consumed} = -\Delta C_A - 2\alpha \frac{\Delta C_A}{C_{source}^2} C_A \qquad (6)$$

In Eqn.6, $\Delta P_{consumed}$ is the amount of budget that can be consumed at A. It consists of two parts: ΔC_A denotes the minimum required budget to go to the next hop, which is the link cost to the next hop. $2\alpha \frac{\Delta C_A}{C_{source}^2} C_A$ is the share of credit that can be used at this hop. Because $\Delta C_A, \alpha$ and C_{source} are constant, the share of credit is proportional to C_A. Thus the higher a node's cost, the more credit it can use.

Although the above analysis is based on a simplified model, it shows quantitatively that nodes closer to the source generally can use more credit than those farther away. Therefore, the forwarding mesh can expand aggressively initially, while still having some credit later to maintain the width. We will explore other forms of threshold function in Section 4.

3.2 Interleaving Paths Add to Robustness

GRAB achieves robustness through the redundancy in the mesh. We carefully make the design choices so that the paths in the mesh are *implicit* and *interleaving*. By *implicit* we mean a sender does not appoint which neighbors should continue forwarding the packet. It is up to each receiver to decide whether or not it should forward. When some neighbors fail or there are packet losses, the sender still performs the same operations. As long as some surviving neighbors receive the packet, they can continue forwarding to the next hop. The lack of explicit path state eliminates the need to repair broken paths.

By *interleaving* we mean these paths are not disjoint, but intersect with each other. This is more robust than multiple disjoint paths, where a single node failure or packet loss destroys the forwarding on a path and each path still has a high probability to fail when there are many hops between the source and the sink. In contrast, interleaving paths recover the node failures and packet losses of each other. For example (Figure 4A), there exist three disjoint paths A1-A2-A3, B1-B2-B3 and C1-C2-C3. If A2 fails and both B3 and C2 do not receive the packet, all three paths fail to deliver

[4] We call the number of nearer neighbors to reach *the branching factor* . It represents a tradeoff between robustness and energy. Experiments and analysis in Section 3.2 show that three is an appropriate number.

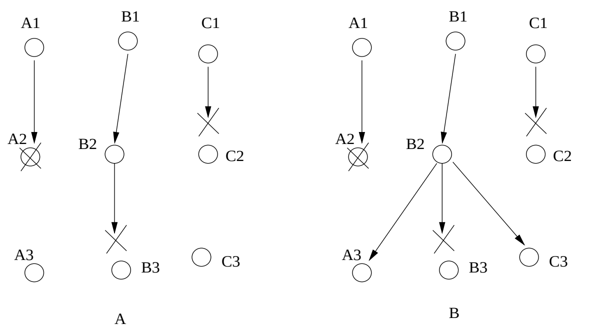


Figure 4: A: any single node failure or packet loss ruins a single path; B: interleaving paths can recover each other from failures or packet losses

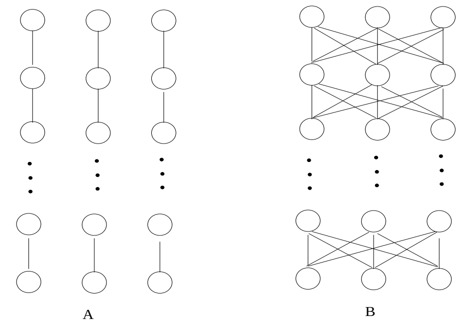


Figure 5: A: forwarding model for disjoint multiple paths B: forwarding model for interleaving paths of GRAB

the packet. In contrast (Figure 4B), given the same failure of A2 and packet losses at B3 and C2, A3 and C3 can still receive from the broadcast of B2. Thus path A and C can be recovered by B2. Similarly, path B can be recovered by broadcasts from A3 or C3 later.

We quantify the robustness against packet losses and node failures using the following model: There are m nodes at each hop; all the m nodes at hop 0 send the packet. In GRAB, each node sends to all the m nodes at the next hop; in multiple disjoint paths, each node sends to only one node at the next hop (illustrated in Figure 5). The probabilities of node failure and packet loss on a link are f, e. We want to derive $P^d_{fail}(N)$, $P^g_{fail}(N)$ the probabilities that the packet fail to reach any node at hop N for disjoint paths and GRAB. Denote the probability that the packet reaches at least one node at hop N as $P_{succ}(N)$. It is obvious that $1 - P_{fail}(N) = P_{succ}(N)$.

First we consider packet losses, assuming $f = 0$. We have

$$P^d_{fail}(N) = (1 - (1-e)^N)^m \tag{7}$$

For an interleaved forwarding mesh, we further define $p(i,j)$ as the probability that i nodes at a hop send the packet and j nodes on the next hop receive it; $F(i,j)$ as the probability that at hop 0 i nodes send the packet, at hop j no node receive the packet and at hop $j-1$ at least one node receives the packet, i.e., the forwarding fails at "exactly" hop j. Thus

$$P^g_{fail}(N) = \sum_{i=1}^{N} F(m,i) \tag{8}$$

We have the following recursive equations

$$\begin{cases} F(i,1) = p(i,0) & i = 0,1,2,...,m \\ F(i,N) = \sum_{j=1}^{m} p(i,j)F(j,N-1) & \end{cases}$$

where

$$\begin{cases} p(0,0) = 1 & \\ p(0,j) = 0 & j \neq 0 \\ p(i,j) = \begin{pmatrix} j \\ m \end{pmatrix} (1-e^i)^j (e^i)^{m-j} & i \neq 0 \end{cases}$$

Given the values of the parameters, we can solve the recursive equations numerically and compare the robustness of GRAB and disjoint paths. Figure 6 plots the $P_{fail}(N)$ for both GRAB and disjoint paths, given that each hop has 3 nodes. For disjoint paths, after 25 hops, more than 90% of the packets are lost. Even a small e of 0.1 does not help much: After 40 hops, more than 95% packets are lost. In contrast, for the interleaved mesh in GRAB, even with $e = 0.4$, only 13% packets are lost after 40 hops. For smaller e's, more than 95% packets can travel as far as 200 hops. The above shows that an interleaved mesh is significantly more robust than disjoint paths.

Increasing m, the number of nodes in a row, can help disjoint paths, but only to quite limited extend. Figure 7 gives the $P_{fail}(N)$ of GRAB and disjoint paths with different m's, given a fixed e of 0.3. When the number of paths increases from 2 to 4, packets do not travel much farther in disjoint paths. After 12 hops, more than 90% of the packets are lost even with 4 disjoint paths. For GRAB, even there are only 2 nodes in a row, it is after 80 hops that the same amount of packets are lost. When one more node is added to each row, the results improve more than a magnitude: 95% of the packets can travel more than 200 hops [5]. When m increases to 4, P^g_{fail} is almost negligible. The above shows that increasing parallelism does not help disjoint paths much, but improves the robustness of an interleaved mesh drastically.

Next we consider node failures without packet loss. We have

$$P^d_{fail}(N) = (1 - (1-f)^N)^m \tag{9}$$

$$P^g_{fail}(N) = 1 - (1-f^m)^N \tag{10}$$

Figure 8 plots the $P_{fail}(N)$ for both schemes under node failure rates (f) of 0.1, 0.2 and 0.3, with the number of nodes in a row (m) fixed at 3. When $f = 0.3$, for disjoint paths, after 9 hops more than 90% packets are lost; for GRAB it is after 85 hops. Comparisons under other node failure rates reveal the same observation: GRAB is magnitudes more robust than disjoint paths under node failures. Similarly, in Figure 9, we fix $f = 0.2$ and vary m from 2 to 4, see how the increased parallelism improves robustness for GRAB and disjoint paths. We have the same observation that increasing m improves the robustness of GRAB drastically but only

[5]This result, together with experiments, is the reason why we choose three as the branching factor.

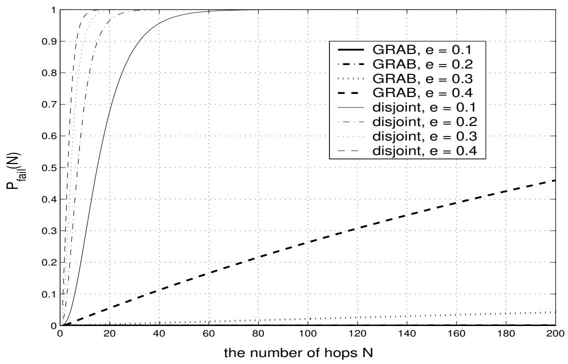


Figure 6: The probability that the packet fails to reach hop N as a function of hop number N, for different packet loss rate e. m is fixed at 3

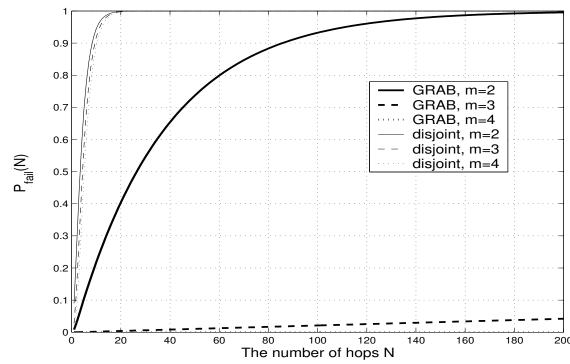


Figure 7: The probability that the packet fails to reach hop N as a function of hop number N, for different number of nodes in a row m. e is fixed at 0.3

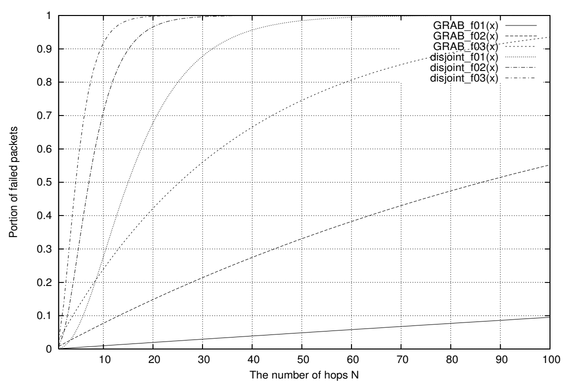


Figure 8: The probability that the packet fails to reach hop N as a function of hop number N, for different node failure rates f, and m is fixed at 3

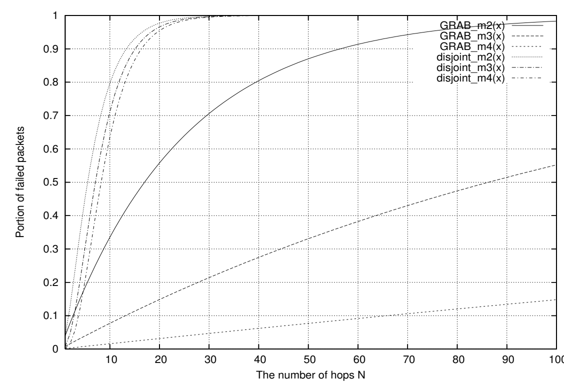


Figure 9: The probability that the packet fails to reach hop N as a function of hop number N, for different numbers of nodes in a row m, and f is fixed at 0.2

slightly for disjoint paths.

The robustness of GRAB, as shown in evaluation (Section 4), is less than the above simple analysis due to a number of factors: Not each node can send to multiple neighbors; packets may collide with each other; the combined effect of node failure and packet loss, etc. Nevertheless, it still demonstrates the advantage of implicit and interleaving paths over disjoint ones.

3.3 Complexities of the Forwarding Algorithm

For each sink, a node A has $O(1)$ state, including its cost C_A to reach the sink, the cost to reach the closest three nearer neighbors. Other states such as C_{source} and the credit α are carried in the packet, not kept by any node. Thus the state complexity is independent of the number of sources. The state complexity is proportional to the number of sinks. When there are n sinks, the state is $O(n)$. Note that the cache is of constant size and independent of the number of sources or sinks.

The computing overhead for each data report is constant. For each report, a node needs to compute both the remaining credit ratio R_α and the threshold R_{thresh}, each of which requires a constant amount of calculation. When there are node failures or packet losses, each node still performs the same operations as before. There is no computation to find new paths to replace broken ones. Thus the total computing overhead is proportional to the number of packets a node has to forward.

The energy overhead consists of two parts, forwarding and cost field building. For each data report, a node sends it only once due to the duplicate suppression. Thus the energy used in forwarding is proportional to the number of data reports. For cost field building, we prove in [16] that a node broadcasts only once for each refreshing of the cost field. Thus this part of energy is proportional to the number of times that the cost field is refreshed and the number of sinks.

4. PERFORMANCE EVALUATION

In this section we evaluate the performance of GRAB through simulations. We implemented GRAB forwarding protocol in Parsec [11] due to its ability to scale to large numbers of nodes. We select sensor hardware parameters similar to Berkeley motes [7]. The maximum transmission range of a node is 10 meters, each node can adjusts its transmitting power to reach a given range. We simulated our test scenarios under both the two ray ground and the free space signal

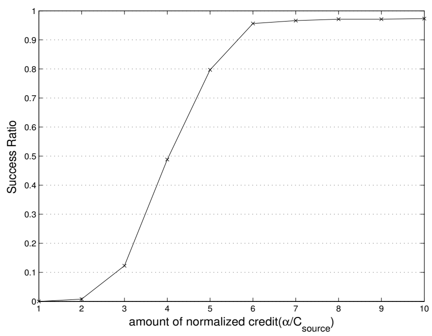


Figure 10: success ratio for different α

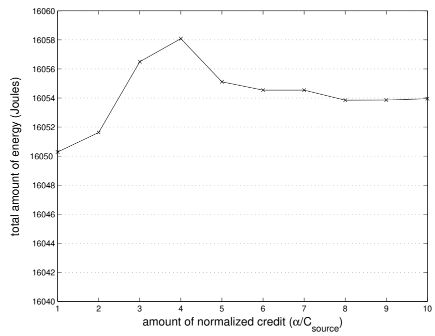


Figure 11: energy consumption for different α

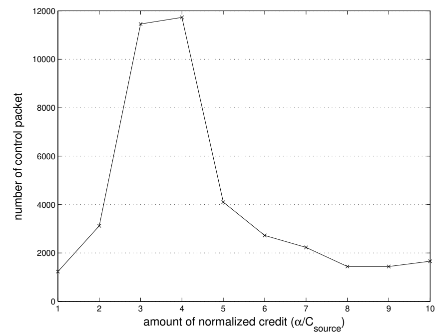


Figure 12: control packet number for different α

propagation models. Due to space limitation we present the results from the former only.[6] The power consumptions of full power transmitting, receiving and idling are 60mW, 12mW and 12mW. The transmission (receiving) time for a packet is 10 ms.

In most scenarios, we use a field size of $150 \times 150 m^2$ where 1200 nodes are uniformly distributed. One sink and one source sit in opposite corners of the field. The source generates a report packet every 10 seconds. In each run 100 reports are generated. The average number of hops of the source' minimum hop path and minimum cost path are about 30, 70, respectively. Packet losses are simulated by a probability of dropping the packet at the receiver. Packet loss rate is defined as the probability. Node failures are uniformly distributed over time. The fraction of failed nodes is defined as the node failure rate.

To test if GRAB achieves its goal in robust data delivery, we measure the *success ratio*, which is the ratio of the number of report packets successfully received at the sink to the total number generated at the source. It indicates the degree of robustness of GRAB to forward data in the presence of node failures and packet losses. To see if GRAB satisfies robustness at the cost of excessive overhead, we also measure *total energy consumption* and *control packet overhead.* Total energy consumption is the total amount of energy consumed in the simulation. It shows how much energy GRAB incurs for robust data delivery. Control packet overhead is the number of control packets in the simulation. The results are averaged over 10 different runs.

We first evaluate the impact of control parameters, including the amount of credit and the threshold function; then the impact of various environmental settings, including node failure rate, packet loss rate, node density and the size of the field. Finally we compare GRAB with Directed Diffusion [8].

4.1 Impact of Control Parameters

4.1.1 *Different Amounts of Credit* α

The amount of credit directly affects the degree of robustness. To find how much credit α is enough for robust delivery, we vary the amount of credit from 1 to 10 times that of the source' cost to reach the sink. A fixed 15% node failure and a fixed 15% packet loss rate are present in all runs.

Figure 10 shows the success ratio as a function of α, which is normalized to the source' cost. When the credit is small, the chance of successful delivery is also very small. When $\alpha \leq 2$, almost all reports are lost. This is because there are many hops (around 70) from the source to the sink, along which node failures and packet losses happen frequently. When α increases, the success ratio improves steadily. $\alpha = 5$ gives an 80% success ratio. When the amount of credit is sufficient, the forwarding is very robust. For $\alpha \geq 6$, over 95% report packets are successfully delivered to the sink[7]. This shows that credit decides the degree of robustness. A sufficient credit ensure good robustness.

To find whether GRAB consumes excessive energy to ensure robustness, Figure 11 gives the total energy consumption as a function of α. When α is small($\alpha = 1$), about 16050 Joules are consumed. As α increases, the total energy also increases. At $\alpha = 4$, total energy reaches 16058 Joules, which is 8 Joules more. This is because more energy is used in data delivery and building the cost field. When $\alpha \geq 6$, the total energy decreases to 16054 Joules. The fluctuation is very small compared to the total amount. Thus GRAB is efficient and does not achieve robustness at the cost of excessive energy consumption.

The decrease of total energy when α is high is a little counter-intuitive because more data packets are successfully delivered and more energy should be used. Actually the decrease comes from the reduced control packets. Figure 12 shows the control packet overhead. When α is small(≤ 2) or big(≥ 6), the delivery quality is constantly low or high. The measured parameters about delivery quality at the sink seldom differ from their recent history beyond the thresholds. Thus less refreshings happen, and less total energy consumption. The number of control packet is below 3100, and on average less than 3 cost field (re)buildings happen. When α is medium(from 3 to 5), the delivery quality is not stable,

[6] Results from the free space model are similar.

[7] A sufficient $\alpha \geq 6$ because we use transmitting energy as the cost. It does not mean 6 times more total energy is consumed. In two-ray ground model, the transmitting power increases linearly to the 4th power of distance. Six times in power means 1.56 times in distance on average. In free space model, $\alpha \geq 1.2$ is sufficient under the same topologies.

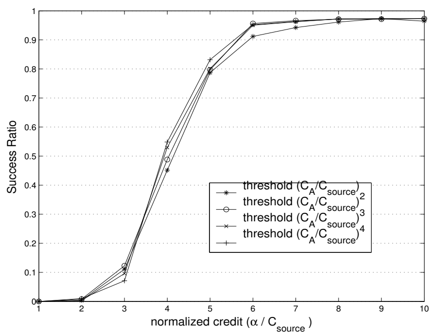


Figure 13: success ratio for different threshold functions

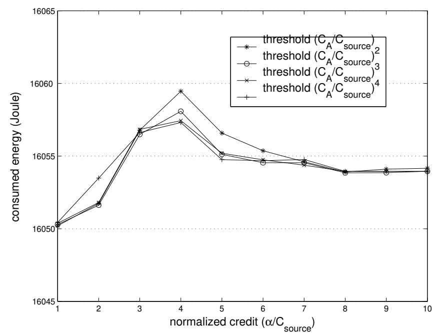


Figure 14: energy consumption for different threshold functions

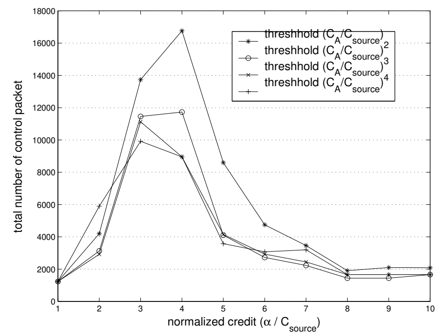


Figure 15: control packet number for different threshold functions

and the measured parameters differ from their recent averages beyond the thresholds more often, triggering more refreshings and thus more energy consumption. The number of control packets reaches 11500, and about 10 cost field (re)buildings happen.

The different control packet overhead also shows that the event-driven cost field refreshing can adapt to the delivery quality. When the delivery quality is stable(either constantly low or constantly high), more refreshings cannot improve the success ratio much(the improvement is less than 1%). So GRAB has less refreshings. When the delivery quality is not stable, the sink refreshes the cost field more often, thus more packets which otherwise could not reach the sink are successfully delivered(the improvement is about 10%).

4.1.2 Different Threshold Functions

The form of the threshold function decides how credit is alloted among different hops. We evaluate four different threshold functions: (C_A/C_{source}), $(C_A/C_{source})^2$, $(C_A/C_{source})^3$ and $(C_A/C_{source})^4$, where C_A is the cost of the receiving node A and C_{source} is the cost of the source to reach the sink. We repeat the same simulations in Section 4.1.1. The success ratio, energy consumption and control packet overhead are shown in Figure 13, 14 and 15, respectively. The success ratios for the threshold (C_A/C_{source}) is smaller than the those of the other three. Its energy consumption and control packet overhead are obviously higher, while the other three have similar energy consumption and control packet overhead. The metrics do not change much for the latter three threshold functions($(C_A/C_{source})^2$, $(C_A/C_{source})^3$ and $(C_A/C_{source})^4$). This is because they all give more credit to beginning hops and still allot some amount to later hops[8]. Thus the forwarding mesh can expand quickly and maintain a certain width later.

4.2 Impact of Environmental Settings

4.2.1 Node Failures and Packet Losses

We evaluate the robustness of GRAB by studying how node failures and packet losses affect the success ratio in this section. We first vary the node failure rate from 5% to 50%, while using a fixed 15% packet loss rate. Then we vary the packet loss rate from 5% to 50%, while using a fixed 15% node failure rate. The amount of credit α is set to 6. This is the value that achieves higher than 95% success ratio in the previous section.

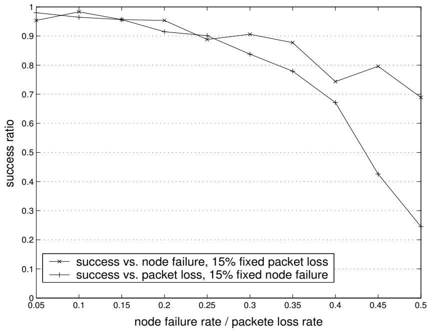
Figure 16 shows the success ratio as functions of node failure rate and packet loss rate. We first look at the impact of node failures. The success ratio is above 95% for node failure rates of up to 20%. As the node failure rate continues to increase, although the success ratio tends to decrease, GRAB still maintains very high degrees of robustness. The success ratio remains above 85% when 35% nodes fail, and is around 70% in the extreme case when half of the nodes fail. This shows that GRAB is robust even with severe node failures. For packet loss rates of up to 25%, the success ratio is above 90%. After 25%, the success ratio drops quickly. With a packet loss rate of 65%, the success ratio is about 67%. Compared to node failure cases, GRAB is less robust when the packet loss rate is high. This is because no acknowledgment or retransmission is used to recover a lost packet in CSMA MAC. For node failures, however, as long as there are still enough surviving nodes, a cost field refreshing can resume data delivery. Nevertheless, GRAB delivers over 80% reports successfully for node failure rates or packet loss rates of up to 30%. The high success ratio also demonstrates that GRAB is highly tolerate to inaccurate cost fields because node failures and packet losses both cause inaccuracies during cost field building.

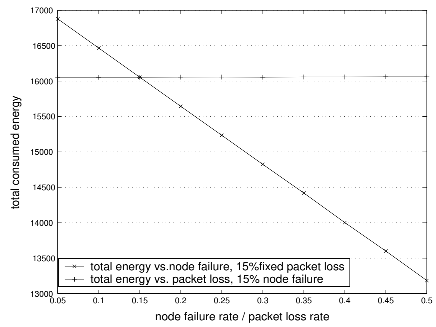
The energy consumptions are shown in Figure 17. When node failure increases, the energy decreases linearly. This is because the idle energy dominates the total energy consumption. A higher node failure rates means more node failures, thus proportionally less energy consumption. For different packet loss rates, the energy remains almost constant around 16054, increasing less than 6 Joules as the packet loss rate grows from 5% to 50%. Again, although less energy is consumed for data delivery, more is spent in rebuilding the cost field. Thus the total energy increases a little.

4.2.2 Impact of Node Density

To find how the density of nodes can affect the robustness of GRAB, we keep the field size $150 \times 150m^2$, while varying the number of nodes from 600 to 1800. The node failure rate and the packet loss rate are both 15%. Figure 18 shows how the success ratio changes over different node numbers. The success ratio is low for node numbers of 600 and 750.

[8]Similar analysis on credit allotment can be made by following the analysis in Section 2.3.2.





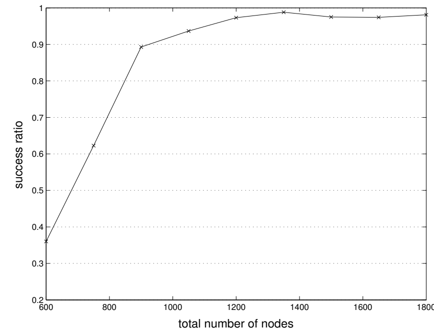


Figure 16: success ratio for node failures and success ratio for packet losses

Figure 17: energy consumption for node failures and energy consumption for packet losses

Figure 18: success ratio for different node densities

The reason for the initial low success ratios is that there are not enough nearer neighbors when node density is low. The variations in topology further exacerbates the situation. Thus the forwarding stops where there are not enough nearer neighbors to combat node failures and packet losses. However, starting from 900 nodes, the success ratio remains high above 90% for all the remaining densities. Since the total energy depends on the number of nodes, we normalize it to the energy consumption per node. It remains constant at around 13.4 Joules.

4.2.3 *Impact of Field Size: Scalability*

We study how well GRAB scales to large numbers of hops for data delivery. We keep the average node density the same while varying the field from $50 \times 50m^2$ to $250 \times 250m^2$. The number of hops between the source and the sink changes in proportion, from about 25 hops to about 135 hops. Both the node failure rate and the packet loss rate are fixed at 15%. We find that regardless of the field size, the success ratio is always around 90%[9]. This demonstrates that GRAB ensures robustness of data delivery over large number of hops.

4.3 Comparison with Diffusion

We also implemented GRAB in ns2 [1] and compared its performance with Directed diffusion. There are 256 nodes in a $1600 \times 1600m^2$ field. A source and a sink sit in opposite corners of the field. Whenever possible, GRAB uses the same parameters as used in ns2 diffusion code. The transmitting, receiving and idling power consumptions are 0.66W, 0.395W, 0.035W, respectively. The source generates a report every half second. One difference is that GRAB uses CSMA for communication[10]. It is less reliable than the 802.11 DCF MAC used by diffusion, which has RTS/CTS/ACK.

We evaluated their performance under three kinds of node failure models: uniform node failure, in which node failures are uniformly distributed over time. Sudden node failure, in which a randomly chosen set of nodes all fail at the same time. This is to simulate some abrupt massive failure. Transient node failure is similar to that in [8]. A randomly chosen set of nodes are turned off for 7 seconds, then another set is chosen and turned off, and so on. This creates lots of dynamics in the network and tests how well the protocols can handle such dynamics.

Due to the space limitation, we only present the success ratio and energy consumption per successful report[11]. GRAB maintains robust data delivery under all three node failure models. Its success ratio is always above or around 90%. Diffusion has high success ratio for uniform node failures. The success ratio decreases when the node failures are sudden. It drops significantly to about 50% when there are transient node failures. This is because diffusion relies on reinforcements to repair broken paths. When there are high dynamics, the paths break more frequently and thus more data are lost. Overall GRAB's robustness is not affected much by different types of node failures, and this robustness is achieved with a less reliable CSMA MAC.

For GRAB, the energy per successful report tends to decrease as more nodes fail. This is because less nodes are alive to forward, thus less energy is consumed. The energy does not change much for all three failure models. Diffusion has almost the same energy consumption as GRAB for the sudden failure model. It consumes slightly more energy for uniform node failures, and significantly more for transient failures. This is because fewer reports are successfully delivered to the sink in the latter two failure models, thus energy consumption per successful report increases. The abnormal energy drop for diffusion with 20% transient nodes failures is because much less reports are delivered and less energy is consumed compared to 16% failure case.

5. RELATED WORK

[9]The only exception is for $50 \times 50m^2$ field, where the variation in topology caused a disconnected source in one run. The average success ratio would be 96% excluding the exception

[10]As mentioned in Section 2.3.2, a sender randomly waits for some time before passing a packet to MAC to reduce collisions

[11]We do not use the total energy consumption here because the two protocols delivers different numbers of successful reports, thus total energy does not reflect the efficiency of them. The per report energy is more meaningful. It tells to successfully deliver a packet, how much energy a protocol needs to consume

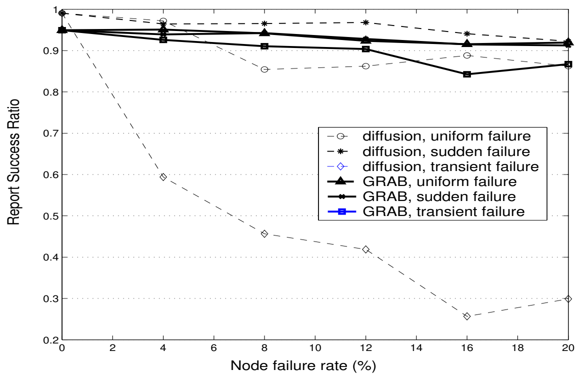


Figure 19: The success ratio of GRAB and diffusion under different node failure models

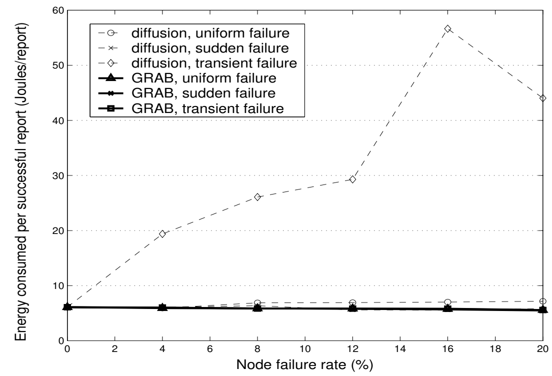


Figure 20: The energy consumption per successful report of GRAB and diffusion under different node failure models

There have been a plethora of research efforts in sensor networking area in the last few years. In this section we describe the basic differences between the existing work and GRAB design.

Directed diffusion [8] is a data forwarding protocol designed for sensor networks where a sink floods its interests to build reverse paths from all potential sources to the sink. GRAB also builds a field, but it is a scalar field of cost values, not one of reverse path vectors. Diffusion uses reinforcement and negative reinforcement mechanisms to select a high quality path for the data flow from each source and deactivate low quality ones. Braided diffusion [5] is a variant of directed diffusion. It maintains multiple "braided" paths as backup. When a node on the primary path fails, data can go on an alternate path. Both Directed diffusion and Braided diffusion establish *explicit* paths to forward data; each node forwards data to a specific next hop neighbor. In contract, a sender in GRAB simply transmits data to the radio channel without appointing any neighbor as the next hop; each receiving node independently decides whether it should further forward the data. There is no explicit path in GRAB; data simply follows whichever surviving nodes to reach the destination.

Diffusion combats against errors and failures by periodically flooding data to repair the paths. GRAB achieves robustness by exploiting the redundancy from interleaving paths in the forwarding mesh. Diffusion detects forwarding loops by caching previous packets. In GRAB, because packets can only go along the decreasing cost direction (toward the sink), no loop can form.

TTDD [18] solves the problem to delivering data to mobile sinks that are in constant motion. It builds a grid structure for each source. The impact of a mobile sink is confined within a local cell. Data delivery and query forwarding traverse the grid tier and the local cell tier in reverse order. TTDD does not address the robustness issue. Only a single path is used to forward data.

Both Diffusion and TTDD work with 802.11 DCF MAC which has RTS/CTS/ACK. They have yet to demonstrate their robustness using nodes with less reliable CSMA MACs.

Cost has been used in other related work. Compared to them, GRAB has the waiting algorithm [16] that builds the cost field in a scalable fashion. Gradient Routing [13] also builds and uses a cost field. But it does not control the degree of redundancy in data forwarding. When a sender broadcasts a packet, all neighboring nodes with lower costs forward the packet. In GRAB, the credit carried in each packet effectively controls the width of the forwarding mesh, thus the degree of redundancy and energy consumption. In Energy Aware Routing [14], the sender picks a neighbor to forward the packet with a probability inverse proportional to its cost. Each packet travels on a different single path, thus the load is balanced among multiple paths. GRAB uses multiple interleaving paths simultaneously, and the sender does not decide which neighbor to continue forwarding. ReInForm [4] uses the cost in another way: Each sender calculates and includes the probabilities for its neighbors to forward the packet when broadcasting. Each neighbor forwards with the corresponding probability. The probabilities control the degree of redundancy to achieve a desired end-to-end success ratio. GRAB uses credit to control redundancy, and without an explicit specification of the end-to-end reliability.

Chen et al. [2] used "ticket", an idea similar to credit, in QoS routing for ad hoc networks to search paths that satisfy delay or bandwidth requirements. The number of tickets carried in a packet decides the number of copies that can be spawn and each copy discover at most one path. Since all paths satisfy the QoS requirement, only one of them is needed for data delivery. The credit of GRAB does not specify the number of paths directly; it translates into the degree of redundancy under the interaction with the cost field. For robustness, multiple interleaving paths are used to forward data in parallel. There is no path state kept in each node about which is the next hop.

The cost field enables greedy forwarding in which the next hop is the neighbor "closest" to the destination. GPSR [10] is a geographical greedy routing protocol that can route packets around "holes" using a planar graph. Compared to that, the cost field in GRAB does not have "holes". As long as a path exist, there is always a neighbor with a smaller cost than the sender at each hop. Also, GRAB uses multiple interleaving paths simultaneously to forward packets.

Redundant mesh forwarding is also proposed in [3, 6] for robust multicast delivery in wireless ad hoc networks. However these designs exchange control messages to establish explicit path state at each node; each node has to maintain the state to identify whether it is within the mesh. In contrast, the forwarding mesh in GRAB is *dynamically* formed by the combined effect of the cost field and the credit value carried in each packet, without keeping any explicit state at each node about whether it is within the mesh.

Routing has been a very active research area in the context of ad hoc networks, many proposals have appeared in the literature [9, 12]. However, they are not designed for sensor networks and do not address the unique issues in sensor networks.

6. CONCLUSIONS AND FUTURE WORK

As the deployment of large scale sensor networks emerges on the horizon today, we are facing new research challenges of providing reliable sensing and robust data delivery via vast numbers of potentially unreliable sensors. Compared to data networks in general, individual sensors have much lower utilization but potentially much higher failure rate. These special requirements demand new solutions to reliable data delivery.

In this paper, we presented the GRAB design which ensures robust data delivery over large numbers of hops of small, unreliable sensor nodes and error-prone wireless channels. GRAB exploits the large scale property of sensor networks and achieves robust data delivery through controlled mesh forwarding. GRAB builds and maintains a cost field for each destination. It controls the "width" of the forwarding mesh, thus the degree of redundancy, by the amount of credit carried in each data packet. Analysis and extensive simulations confirmed GRAB's effectiveness in providing reliable delivery under severe operational conditions, demonstrating the principle that a reliable system can be built out of unreliable components.

We plan to further improve GRAB to make the credit assignment adaptive. The sink may include some information that reflects recent data delivery quality when sending ADV packets. A source can use this feedback to choose an appropriate credit to adapt to network conditions. In addition, the allotment of credit among different hops can also adapt to local failure and noise characteristics. Nodes in a neighborhood with more severe conditions can use greater shares of the credit if they can measure local failures or packet losses.

So far we have been focusing on one stationary sink. When there are multiple sinks, each needs to build its own cost field. Every node keeps one cost per sink. This per-sink state may not allow GRAB to scale to large numbers of sinks directly. Sink mobility is not well addressed in the current design, either. The straightforward solution of re-building the cost field whenever the sink moves may consume excessive energy and bandwidth. Actually, only when the sink moves far away from its location is a totally new cost field needed. If the mobility is low, the sink may leave a "trace" as it moves, by installing state in intermediate nodes. Data reaching its old location follow the trace to reach the mobile sink. Another possibility is to set up relatively stable landmarks [15] and each mobile sink maintain data forwarding paths from the nearest landmark. Data are attracted to landmarks first, then distributed toward mobile sinks. We plan to further investigate multiple, mobile sinks in the future.

7. REFERENCES

[1] Ns2 network simulator. `http://www.isi.edu/nsnam/ns/`.

[2] S. Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad-Hoc Networks. *IEEE Journal of Selected Areas in Communications*, 17(8), 1999.

[3] C.-C. Chiang, M. Gerla, and L. Zhang. Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks. *Cluster Computing*, 1(2):187–196, 1998.

[4] B. Deb, S. Bhatnagar, and B. Nath. ReInForM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks. In *28th Annual IEEE Conference on Local Computer Networks*, 2003.

[5] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *ACM Mobile Computing and Communications Review, Vol. 5, No. 4*, October 2001.

[6] J. J. Garcia-Luna-Aceves and E. L. Madruga. A multicast routing protocol for ad-hoc networks. In *INFOCOM (2)*, pages 784–792, 1999.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, 2000.

[8] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. *ACM International Conference on Mobile Computing and Networking (MOBICOM'00)*, 2000.

[9] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad-hoc Wireless Networks. *Mobile Computing, Kluwer Academic Publishers*, 1996.

[10] B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. *ACM International Conference on Mobile Computing and Networking (MOBICOM'00)*, 2000.

[11] U. Parallel Computing Laboratory, Computer Science Department. Parsec. `http://pcl.cs.ucla.edu/projects/parsec/`.

[12] C. Perkins. Ad-Hoc On Demand Distance Vector Routing (AODV). *Internet-Draft*, November 1997.

[13] R. Poor. Gradient Routing in Ad Hoc Networks. `http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf`.

[14] R. C. Shah and J. Rabaey. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. *WCNC*, 2002.

[15] P. F. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large networks. *Computer Communication Review*, 18(4), August 1988.

[16] F. Ye, A. Chen, S. Lu, and L. Zhang. A Scalable Solution to Minimum Cost Forwarding in Large Scale Sensor Networks. *The Tenth International Conference on Computer Communications and Networks*, 2001.

[17] F. Ye, S. Lu, and L. Zhang. GRAdient Broadcast: A Robust, Long-lived Large Sensor Network. `http://irl.cs.ucla.edu/papers/grab-tech-report.ps`, 2001.

[18] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *ACM International Conference on Mobile Computing and Networking (MOBICOM'02)*, 2002.

Appendix

Pseudo code for GRAB forwarding algorithm.

Forward a received packet REP
/* comparing its own cost to that of the sender */
if($C_{receiver} \geq REP.C_{sender}$)
    /* drop the packet if its cost is higher or equal */
    drop the packet;
else {
    /* calculate how much credit has been used */
    $\alpha_{used} = REP.P_{consumed} + C_{receiver} - REP.C_{source}$
    /* calculate remaining credit ratio */
    $R_\alpha = \frac{REP.\alpha - \alpha_{used}}{REP.\alpha}$
    /* calculate the threshold */
    $R_{thresh} = (\frac{C_{receiver}}{REP.C_{source}})^2$
    /* forward the packet if remaining is greater */
    if($R_\alpha \geq R_{thresh}$) {
        /* increase the total consumed cost by how */
        /* much this node uses to broadcast REP */
        $REP.P_{consumed} = REP.P_{consumed} + P_{thishop}$
        include its own cost in REP;
        broadcast REP to reach three nearer neighbors;
    }
    else {
        update $REP.P_{consumed}$ similarly;
        include its own cost in REP;
        broadcast REP to reach only next hop neighbor
    }
}

Table 1: Pseudo code for forwarding