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

Abstract. Recent technology advances in low-cost, low-power chip designs have made feasible the deployment of large-scale sensor networks. Although data forwarding has 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 spite of unreliable nodes and fallible wireless links. Similar to previous work [1], GRAB builds and maintains a cost field, providing each sensor in the network the direction to forward sensing data. Different from all the existing approaches, however, GRAB forwards data along an interleaved mesh from each source to the receiver. The width of the forwarding mesh is controlled by the amount of credit carried in each data message, allowing the degree of delivery robustness to be adjusted by the sender. 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. As demonstrated in our extensive simulation experiments, GRAB can successfully deliver above 90% of data with relatively low energy cost even under adverse conditions of up to 30% node failures compounded with 15% link packet losses.

1 Introduction

Recent technology advances in low-cost, low-power chip designs have made it economically feasible to deploy large-scale sensor networks. Thousands or even millions of small, inexpensive, and low-power sensors, such as Berkeley Motes[2], can be quickly deployed to monitor a vast field. The sensors collectively sense the environment and deliver the sensing data via a wireless channel. In near future such sensor networks may play an important role in both civil applications such as agriculture as well as disaster recovery and military surveillance. On the other hand, the above mentioned potential applications also present great challenges to reliable sensing data delivery. Wireless communications among the small, power-limited sensor nodes are prone to errors. Severe operational conditions (e.g. strong wind or high temperature) and disasters (e.g. fire or earthquake) may easily destroy individual sensors, resulting in a constantly changing topology. Furthermore, the short transmission range of small sensors also means that sensing data may travel through a large number of hops to reach intended destinations, with potential delivery 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 point (called the *sink*) using unreliable sensor nodes with error-prone wireless links. The objects or events to be monitored are called *stimuli*. All the sensor nodes that detect the same stimulus collectively select the one with strongest sensing signal to generate a sensing report. We call such a node a data *source*. Although several data forwarding protocols have been designed for sensor networks, such as Directed Diffusion [3] and TTDD [4], they typically assume a relatively stable sensor network where nodes do not fail frequently and unexpectedly.

GRAB achieves robust data delivery through building and maintaining a *cost field* for the sink. Each node keeps a cost for forwarding a packet along a certain path to the sink. Nodes "closer" to the sink have smaller costs. A packet can follow the direction of decreasing cost to reach the sink. In stead of a sender appoints which receivers to continue forwarding, GRAB lets each receiver decides whether it should forward by comparing its cost to that of the sender. Multiple such paths exist between a source and the sink.

To further control the redundancy of the multiple paths, a source assigns a *credit* to the packets it sends out. The credit is some extra budget that allows multiple copies of a packet be forwarded over a mesh of *interleaved* paths, each of which has a cost not greater than the total budget. 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 ones. A packet is forwarded over multiple paths, which improves reliability. Such paths interleave and recover each other from node failures or link errors, further increasing robustness. Since it is the receivers, not the sender that decide which nodes should forward, a sender merely broadcasts a packet without worrying repairing failed nodes or broken links. The packet is delivered to the sink by those surviving nodes. This receiver-based design eliminates the overhead of repairing paths of failed nodes or broken links. The credit provides a means to trade off between robustness and total cost. A source can assign a credit that achieves required robustness without causing excessive redundancy.

The rest of the paper is organized as follows: We present the design of GRAB in Section 2. Then we evaluate its performance in Section 3. We discuss future work to GRAB in Section 4. In Section 5, we first describe the differences between GRAB and existing work in sensor networking area, then report our performance comparison study of GRAB with an existing protocol [3]. Section 6 concludes the paper.

2 GRAB Data Forwarding Protocol

2.1 Design Overview

In this paper we assume the following sensor network model: Large numbers of small, stationary sensor nodes are densely deployed over a field. A stimulus is detected by multiple nearby sensor nodes for reliable sensing. Nodes are equipped with CSMA MACs. The lack of RTS/CTS/ACK makes packets more easily lost than those sent with 802.11 DCF. External noises and disturbances may further exacerbate the condition. Sensor nodes fail unpredictably due to the harsh environment. 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]. We use an example of one sink and one stimulus to illustrate how GRAB works.

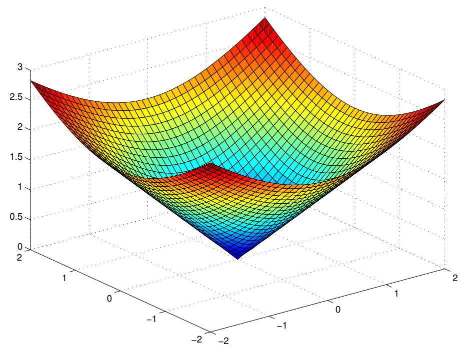
To collect data reports, the sink first builds a *cost field* by propagating advertisement (ADV) packets in the network. The cost at a node is the minimum energy overhead to forward a packet from this node to the sink along a path. We assume each node can estimate the cost of sending data to nearby neighbors. 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 for a illustration): 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 sends a packet, it does not designate which nodes are the next hop. It just includes its own cost in the packet and broadcasts 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. The paths of decreasing cost interleave and form a mesh.

The selection of the 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).

[1] Some existing hardware [2] already have different levels of transmitting power.

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



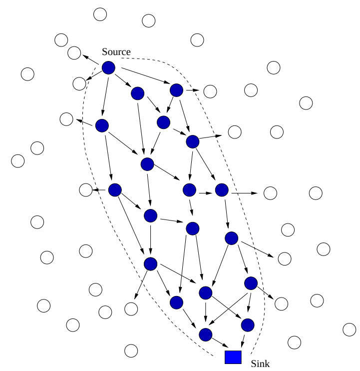


Fig. 1. The "shape" of the cost field is like a funnel, with the sink sitting at the bottom. Packets follow the decreasing cost direction to reach the bottom of the cost field, which is the sink

Fig. 2. The forwarding mesh starts from a source and ends at a sink. All black nodes within the mesh participate data delivery and forward the packet to the sink collectively. Notice that some nodes outside of the mesh also receive the packet but do not forward it.

COS election and data forwarding utilize the same concept of a funnel-shaped field. 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.

After the cost field is built, 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. Ideally, the mesh starts at the source and expands to a certain width quickly, then it keeps the width while going towards the sink until finally it reaches the sink (see Figure 2 for an example). The width of the forwarding mesh determines the robustness of data forwarding.

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 surround 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.

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 a simple model of one stationary source (COS) and one stationary sink. However we point out that the GRAB supports data forwarding from multiple, mobile *stimuli* as well. When a stimulus, such as a tank, moves through the field, 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. For multiple stimuli multiple COS's are elected. The exact details of COS election are not presented in this paper, which focuses on data forwarding.

In the rest of this section, we give a brief summary in Section 2.2 about an algorithm proposed in a previous work [5] to build the cost field efficiently. Then we present the GRAB forwarding algorithm in Section 2.3.

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 obtains a cost by adding the link cost to the cost of the sender. It compares this cost to its old one and sets the new cost as the smaller of the two. Whenever it obtains a cost smaller than the old one, it also 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 [5] and proved that the waiting algorithm ensures each node broadcasts only once, and with its minimum cost.

The value of 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 3), 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 numbered so a sink can calculate success ratio), the average consumed budget, the average number of copies received per packet and the average number of hops traveled 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,

[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 [3]

the sink broadcasts a new ADV packet to rebuild the cost field. Due to space limit, more details are in a technical report[6].

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 how much change 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 minor changes, which will be shown in Section 3.

Before we proceed to the forwarding algorithm, we want to point out that [5] 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 Realizing a Robust Forwarding Mesh by the Credit

This section describes how to build a forwarding mesh by the credit. To realize the mesh we need to address three issues.

Issues in Realizing the Mesh First, how to expand the mesh 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 can fail before the mesh is wide enough.

Second, after the mesh has expanded sufficiently, how to maintain the width. Due to node failures and packet losses, the number of parallel nodes that forward a packet tend to decrease from one hop to the next. A failed node or a node that does not receive the packet can reduce the number of parallel forwarding nodes. 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 are 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.

Solutions to the Issues To address the first two issues, we divide the total amount of credit among different hops in the right way. Specifically, we want beginning hops to consume larger shares of the credit, while later hops consume some, but smaller shares of the credit. 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 should 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 and 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 maintain the width. Otherwise node failures and packet losses can "narrow down" the mesh.

Now, how to solve the last issue? If a packet has been traveling on a quite devious path, or divert from the direction of the sink too much, it would 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 how far it still ahead we can detect and terminate such packets.

To calculate the remaining credit, we let each packet carry certain information, so a node can first calculate how much credit has been consumed. To tell how "far" a node is to the sink, it uses a threshold function, whose value tells the relative "position" of this node between the source and the sink. By comparing the remaining credit to the threshold value, we can tell if the packet has already consumed too much credit, or there is still enough to use. We choose the format of the threshold function such that it achieves desired division of credit among different hops, solving the first two issues (an analysis will be shown later). We first explain what are carried in a packet, then present the detailed forwarding algorithm.

The Forwarding Algorithm A packet carries the following fields:

- α: the amount of credit assigned to the packet at the source. A node needs it to calculate how much credit remains. 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 set to the cost used by the source to broadcast the packet initially and increased by the amount used to forward the packet at each hop.

After a COS assigns an α to a data report, it fills the above three fields and broadcasts the packet. To prevent loops, only receivers with smaller costs may forward the packet. Thus a packet is forwarded by successive nodes of decreasing costs, leading to the sink finally.

If a receiver finds it has a smaller cost, it calculates and compare two ratios R_{α} and R_{thresh} as follows.

$$R_{\alpha} = \frac{REP.\alpha - \alpha_{used}}{REP.\alpha} \tag{1}$$

$$R_{thresh} = \left(\frac{C_{receiver}}{REP.C_{source}}\right)^2 \tag{2}$$

where

$$\alpha_{used} = REP.P_{consumed} + C_{receiver} - REP.C_{source} \tag{3}$$

In the above equations, α_{used} stands for the amount of credit that has been consumed. REP is the report packet received, $C_{receiver}$ is the cost of this node. $REP.P_{consumed} + C_{receiver}$ is the least amount of total budget required should this node forward the packet via any path to the sink. This minimum amount is achieved when the packet took the minimum cost path from this node to the sink. The "extra" amount of this to C_{source}, would be the credit consumed. So Eqn.3 is at least how much credit has been used. Thus R_α represents the fraction of credit that is still available for this node and later hops. R_{thresh} indicates how "far" the node is to the sink. Both R_α and R_{thresh} 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*). How much 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[5]. It increases $P_{consumed}$ by how much budget it is going to consume in broadcasting. Then it broadcasts the packet to reach those nearer neighbors. Different forwarding nodes on the same hop may reach the same nearer neighbor(s) on the next hop. This is how the paths interleave.

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 increases $P_{consumed}$ in the sent packet similarly.

To reduce collisions, a forwarding node always waits for some random time before sending the packet, so that senders on the same hops 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.

Analysis of Credit Allotment Now we give an analysis of the amount of credit that can be used at any hop. For a node A, its cost is C_A. The maximum

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

share of credit is consumed when the remaining credit ratio R_α is equal to threshold R_{thresh}, i.e.

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

Taking derivatives of C_A 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:

$$\Delta P_{consumed} = -\Delta C_A - 2\alpha \frac{\Delta C_A}{C_{source}^2} C_A \tag{5}$$

In Eqn.5, $\Delta P_{consumed}$ is the amount of cost that can be consumed at A. ΔC_A denotes the minimum required cost 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 roughly the maximum amount of credit that can be used at this hop It is proportional to C_A, the cost from this node to the sink. Thus the higher a node's cost, the more credit it can use. Therefore, as a packet travels from source to sink, it is allowed to consume more credit near the source, and less at later hops. This way, the forwarding mesh can expand aggressively initially, while still having some credit later to maintain the width. We will evaluate other forms of threshold function in Section 3.

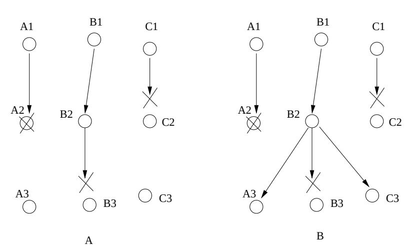


Fig. 3. A: any single node failure or packet loss ruins a single path; B: interleaving paths can recover each other from failures or packet losses

Implicit and 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*. Implicit means a sender does not appoint which node should continue forwarding. It is up to each receiver to decide whether or not it should forward. When there are node failures or packet losses, each node still perform the same operations. As long as there are some surviving nodes that can continue forwarding the packet, data delivery will not fail. The lack of explicit paths eliminates the need to repair them when they are broken.

Interleaving means these paths are not disjoint, they intersect with each other. This is more robust than multiple disjoint paths. The failure of any single node or loss of packet along a single path destroy the forwarding on the path. When there are many hops between the source and the sink, a single path has a high probability to fail. In contrast, interleaving paths in a forwarding mesh can recover the node failures and packet losses of each other. For example (Figure 3A), 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 the packet. In contrast(Figure 3B), given the same failure of A2 and loss of packet 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.

3 Performance Evaluation

In this section we evaluate the performance of GRAB through simulations. We implemented GRAB forwarding protocol in Parsec [7] due to its ability to scale to large numbers of nodes. We select sensor hardware parameters similar to Berkeley motes [2]. The maximum transmission range of a node is 10 meters, each node can adjusts its transmitting power to reach a given range. We simulated both the two ray ground and the free space signal propagation models. Due to space limit we present the results from the former only[5]. 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 cost path is about 70 hops. To simulate fallible wireless links, packets are dropped at the receiver with a probability, which is called packet loss rate. 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.

[5] Results from the free space model are similar

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.

3.1 Impact of Control Parameters

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.

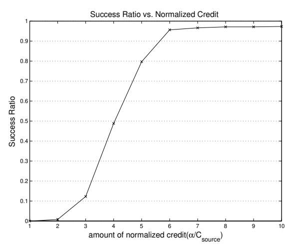


Fig. 4. success ratio for different α

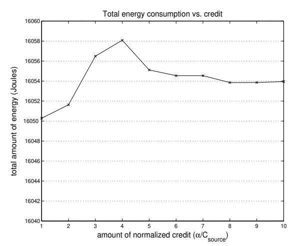


Fig. 5. energy consumption for different α

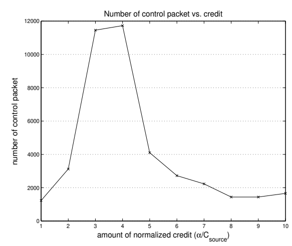


Fig. 6. control packet number for different α

Figure 4 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[6]. 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 5 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

[6] 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.

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 6 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, 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%).

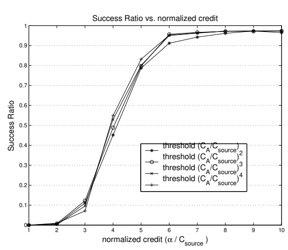


Fig. 7. success ratio for different threshold functions

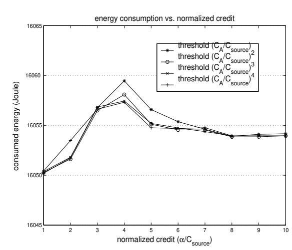


Fig. 8. energy consumption for different threshold functions

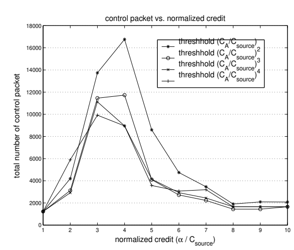


Fig. 9. control packet number for different threshold functions

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 3.1. The success ratio, energy consumption and control packet overhead are shown in Figure 7, 8 and 9, 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[7]. Thus the forwarding mesh can expand quickly and maintain a certain width later.

3.2 Impact of 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.

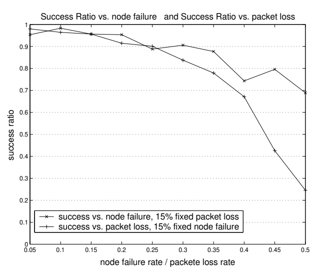


Fig. 10. success ratio for node failures and success ratio for packet losses

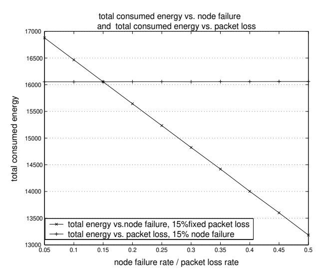


Fig. 11. energy consumption for node failures and energy consumption for packet losses

Figure 10 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 acknowledgement 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

---

7 Similar analysis on credit allotment can be made by following the analysis in Section 2.3.

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 11. 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 Future Work

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. Although a sink can simply rebuilds its cost field every time it moves to a new location, such rebuildings may consume much energy and bandwidth when the sink is highly mobile. We plan to apply landmark routing [8] to address the multiple, mobile sink problem in the future.

5 Related Work

There have been a plethora of research efforts in sensor networking area in the last few years. Directed diffusion [3] 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 [9] 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 decides whether it should further forward the data. There is no explict path in GRAB; data simply follows whichever surviving nodes to reach the destination.

Diffusion combats against errors and failures by periodically re-flooding the interests messages 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 [4] 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.

Gradient Routing [1] shares similarity in design with GRAB in that it also builds and uses a cost field. However it has no mechanism to control the degree of redundancy in data forwarding. When a sender broadcasts a packet, all neighboring nodes with lower costs forward the packet, leading to much redundancy and higher energy consumption. In GRAB, the credit carried in each packet effectively controls the width of the forwarding mesh, thus the degree of redundancy and energy consumption.

Energy Aware Routing [10] also builds per-sink cost fields to direct data delivery but it uses single path only. A sender probabilistically pick a receiver to forward the packet. GRAB has multiple interleaving paths forming a mesh and senders do not decide who are receivers.

Redundant mesh forwarding is also proposed in [11, 12] for robust multicast delivery in wireless ad hoc networks. However these designs exchange control messages to establish explicit path states at each node; the forwarding mesh is made of a set of explicit paths. 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, which allows data to flow along any path 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 [13, 14]. However, they are not designed for sensor networks and do not address the unique issues in sensor networks.

6 Conclusions

As the deployment of large scale sensor networks showing up 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. 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.

References

1. Poor, R.: Gradient Routing in Ad Hoc Networks. (http://www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf)
2. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System Architecture Directions for Networked Sensors. International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX) (2000)
3. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. ACM International Conference on Mobile Computing and Networking (MOBICOM'00) (2000)
4. Ye, F., Luo, H., Cheng, J., Lu, S., Zhang, L.: A two-tier data dissemination model for large-scale wireless sensor networks. In: Mobicom. (2002)
5. Ye, F., Chen, A., Lu, S., Zhang, L.: A Scalable Solution to Minimum Cost Forwarding in Large Scale Sensor Networks. The Tenth International Conference on Computer Communications and Networks (2001)
6. Ye, F., Lu, S., Zhang, L.: GRAdient Broadcast: A Robust, Long-lived Large Sensor Network. http://irl.cs.ucla.edu/papers/grab-tech-report.ps (2001)
7. Parallel Computing Laboratory, Computer Science Department, U.: Parsec. (http://pcl.cs.ucla.edu/projects/parsec/)
8. Tsuchiya, P.F.: The landmark hierarchy: A new hierarchy for routing in very large networks. Computer Communication Review **18** (August 1988)
9. Ganesan, D., Govindan, R., Shenker, S., Estrin, D.: Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. ACM Mobile Computing and Communications Review, Vol. 5, No. 4 (October 2001.)
10. Shah, R.C., Rabaey, J.: Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. WCNC (2002)
11. Chiang, C.C., Gerla, M., Zhang, L.: Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks. Cluster Computing **1** (1998) 187–196
12. Garcia-Luna-Aceves, J.J., Madruga, E.L.: A multicast routing protocol for ad-hoc networks. In: INFOCOM (2). (1999) 784–792
13. Johnson, D.B., Maltz, D.A.: Dynamic Source Routing in Ad-hoc Wireless Networks. Mobile Computing, Kluwer Academic Publishers (1996)
14. Perkins, C.: Ad-Hoc On Demand Distance Vector Routing (AODV). Internet-Draft (November 1997)