Energy Efficient Routing with Guaranteed Delivery in Wireless Sensor Networks

Di Tian
Multimedia Communications Research Laboratory
School of Information Technology and Engineering
University of Ottawa
Ottawa, ON, Canada
1-613-5626800x2148
dtian@site.uottawa.ca

Nicolas D. Georganas
Multimedia Communications Research Laboratory
School of Information Technology and Engineering
University of Ottawa
Ottawa, ON, Canada
1-613-5626800x6225
georganas@discover.uottawa.ca

***Abstract*— High delivery ratio with low energy consumption is one of design challenges for wireless sensor network routing protocols. In this paper, we identify the drawbacks of pure single path routing scheme and multi-path routing scheme, in terms of guaranteed delivery with low energy consumption. Accordingly, we describe a scheme, in which data is forwarded along a pre-established single path to save energy, and a high delivery ratio is achieved by path repair whenever a break is detected. We propose a simple, quick, local path repairing approach, whereby a pivot node can skip over path break by only using the already existing routing information in its neighborhood. We implement this scheme and compare its performance with those of pure single path without repair and two multi-path routing schemes. Simulation results show that in the same network topology with the same failed nodes, our single-path with repair routing scheme (SWR) has the highest delivery ratio among all the compared schemes in almost all the test cases. The energy consumed by SWR for one data delivery is relatively low compared with the other schemes.**

Keywords-reliable delivery, energy efficiency, wilress sensor network

I. INTRODUCTION

A wireless sensor network consists of a large number of tiny sensing devices, deployed in a region of interest. Each device has processing and wireless communication capabilities, which enable it to gather information from the environment and to generate and deliver report messages to the remote base station (or sink node). The base station aggregates and analyzes the report messages received and decides whether there is an unusual or concerned event occurrence in the area of interest [1-4]. Unlike Mobile Ad Hoc networks, wireless sensor networks are characterized by asymmetric many-to-one data flows (mainly from sensor nodes to sink node), severe energy constraints and unreliable network nodes. Therefore, most routing protocols proposed for Mobile Ad Hoc networks are not suitable for wireless sensor networks, or cannot be used in wireless sensor networks without any modification. Thus, alternative approaches need to be explored. The main challenge of wireless sensor network routing protocols is to achieve maximal robustness against path failure with minimal energy consumption.

Existing major routing protocols for wireless sensor networks include *LEACH* [11], *Directed Diffusion* [8], *Energy Aware Routing* [9], *Rumor* [12], *Braided* [6], *MESH* [5] etc. *LEACH* is built on the assumption that all sensor nodes can reach the sink node directly. Therefore, it is only applicable for networks with small geographical size. Except *LEACH*, all the other protocols support multi-hop routing. Depending on how many copies of one data packet are forwarded to the destination simultaneously, these multi-hop routing protocols can be divided into two categories: single-path routing and multi-path routing. In single-path routing, for each data packet, there is only one copy traveling along one path in the network. While in multi-path routing, multiple copies of one packet are transmitted in parallel along different paths to the same destination. Among the above-mentioned multi-hop protocols, only *MESH* is explicitly claimed as multi-path routing. *Braided* builds multiple paths for a data delivery, but only one of them is used, while others are maintained as backup paths. *Directed Diffusion* can be single-path or multi-path routing depending on how many paths are reinforced by sink node. *Energy aware routing* and *Rumor routing* are single-path ones.

Generally, single path routing is simple and consumes less energy than multi-path routing. However, a single path failure will cause a break of transmission and hence completely ruin the delivery. Compared with other wireless networks, wireless sensor networks are subject to high node failure rate P_n and high channel error rate P_l. Also, they have a relatively short radio range and may be deployed into a large geographical coverage area, i.e., the route between a source node and a sink node is quite likely to consist of a large number of hops L. As a result, the success probability provided by single path routing, which is given as $(1-P_n \cup P_l)^L$, is very low as shown by our experimental results later.

Consequently, more and more researchers are resorting to multi-path routing for delivery success. For instance, sending the same data packet along two fully node-disjointed paths (if they exist) almost doubles the delivery ratio. Using k-fully node-disjointed paths (k>2) can further increase the delivery ratio in approximate proportion to k. Moreover, if we relax the requirement for disjointed-ness, partial or interwoven

multi-path routing schemes have shown higher resilience to single path failure theoretically and experimentally [4]. However, determining the width of multi-path routing (for instance, the value of k in k-fully node-disjointed paths) before transmission is not so easy, because sensor network topologies often change unpredictably due to sudden node malfunction, environmental physical damage and impulsive strong external interference. Large k values can ensure success of deliveries, but may cause unnecessary energy waste. In contrast, a small k value saves energy, but may not guarantee the highly demanded delivery ratio. Another disadvantage of a large k value is that: the larger the k value is, the more traffic is generated for one data packet delivery, which may cause network congestion. Given that the simplest CSMA scheme is used at the MAC layer, more traffic means a longer backoff delay waiting for transmission and more collisions induced in the wireless channel. Unless the source nodes are notified of path quality in a certain way, it is impossible to adjust the optimal k value dynamically to adapt to unpredictable network topology changes.

One intuitive approach for achieving a high delivery ratio with low energy consumption is to forward data along a single path and to repair the path whenever a break is detected. Path repair has been introduced into many wireless routing protocols [13, 14]. Depending on how the original path is established and what is the reason of path break, repair approaches are different. In [13, 14], when a path break (failure) is detected, a notification is sent to the source node, which is responsible for finding an alternative path and resending the data packet. However, this kind of source-initiated path repairing approach is uneconomical, especially when a failure occurs many hops away from the source node. In this work, we propose a local pivot-initiated path repairing approach, whereby the node (called pivot node), which is located at the immediate upstream of a path break, is responsible for seeking alternative paths through a local survey. If alternative paths exist, data forwarding will proceed along the best of them without restarting from the source node. Although the selected alternative path may not be optimal from the view of the source node, the energy is conserved by preventing the previous transmission effort from being wasted, avoiding long-distance failure notification, and restricting the range of alternative path seeking into a small local area. Such energy saving should outweigh the additional energy caused by using a non-optimal path.

In this paper, we describe and evaluate a single-path with local impromptu repair data forwarding scheme. We organize the paper as follows. In section II, we explain how to establish an optimal path in the proposed scheme and how to forward data along the optimal path and how to detect a broken link during delivery. In section III, we describe the impromptu local path repair approach in details. In section IV, we evaluate the performance of the proposed scheme and compare it with those of three other schemes. Section V presents conclusions and future work.

II. DATA FORWARDING ALONG OPTIMAL PATH

A. Optimal Path Setup

In the proposed scheme, before data transmission, an optimal path from each sensor node to each sink node needs to be established proactively, which can be accomplished through a global lowest-cost path setup process initiated from each sink node. A detailed description and discussion about the straightforward flood-based path setup approach and the more energy efficient backoff-based approach can be found in [5]. Because the path establishment is not core of this paper and the space is limited, we directly use the result of the path setup process presented in [5]. That is after it, each node stores the following information in its routing table: cost C (the cost from the current node to the sink node along the lowest-cost path), downstream node Id $ID_{downstream}$ (the next hop from the current node along the lowest-cost path) and downstream node cost $C_{downstream}$ (the minimal cost of the downstream node to the sink node).

Note that cost can be the hop number, energy consumption, delay, remaining energy, etc. In this paper, we use the hop number as the metric of cost because it does not need any extra explanation. Furthermore, when sensor nodes have a fixed transmission range, the hop number is also the indicator of energy consumption.

B. Data Forwarding Along the Optimal Path

When a data packet needs to be delivered to a sink node, $ID_{downstream}$ recorded in the routing tables of the source node and all its successors will direct the packet to flow down from the source node along its optimal path to the destination.

Each data packet carries the following fields in its header: *source_id*, *seq_#*, *sender_id*, *sender_cost*, *return_num*, *return_limit* and *direction*. *seq_#* is associated with the source node and incremented each time the source node issues a new data packet. The combination of *source_id* and *seq_#* is used to uniquely identify a data packet. *Direction* is a one-bit binary field. *0* indicates that the data packet is forwarded to the sender's downstream node (successor), while *1* indicates that the data packet is returned to the sender's upstream node (predecessor). In the later description, we call the data packet with 0 in the *direction* field as "*forwarded data packet*"and that with 1 in the *direction* field as "*returned data packet*". *sender_id* tags the transmitter Id. *sender_cost*, *repair_num*, and *repair_limit* are associated with path repair. Their purpose will be explained in the next section.

After first receiving a forwarded data packet, a node is required to record the values of *source_id*, *seq_#* and *sender_id* into its data cache, fill its own id into the *sender_id* field, and then forward the updated data packet to its next hop, which is indicated by $ID_{downstream}$ in its routing table.

C. Detecting the Broken Llink

When a data packet is delivered along the lowest cost path, any node failure or channel error along the path may

cause a break in transmission. To guarantee the delivery, each node, which forwarded a data packet, is responsible for confirming that its successor has successfully received the packet. This may be implemented by the transmitter monitoring the packet just sent out to the downstream node and overhearing if that node has passed it on within a time period T_f. Of course, if link level acknowledgement is supported by the MAC layer protocol (for instance, 802.11 has such function), the above passive acknowledgement scheme is unnecessary. If neither of the above confirmation mechanisms is available, the transmitter may set a bit in the header of data packet to request explicit end-to-end acknowledgement from the next hop. In any way, the transmitted data packet has to be kept in the buffer before its receipt has been confirmed.

A failed receipt may result from either or both of two reasons: node failure or channel error. Channel error refers to temporary path break due to a collision, interference or obstacle in the wireless channel. In contrast, node failure refers to permanent path break due to energy exhaustion, malfunction or physical damage of sensor nodes. According to [4], channel error is supposed to be solved through ARQ or FEC mechanism at the data link layer. While in [15], it is dealt with at the transport layer. In order to evaluate the performance of the proposed scheme in the worse case, however, in this paper, we assume that there are no any defense functions against channel error at the other layers. Moreover, to avoid extra communication overhead, we don't introduce any mechanisms to determine if a failed receipt is caused by a node failure or channel error. Therefore, we deal with both kinds of transmission break in the same way as described in the next section.

III. IMPROMPTU LOCAL PATH REPAIR

As described in the previous section, each intermediate node is responsible for confirming that its successor has successfully received the data packet it just sent out. When the transmitter detects a failed receipt, it assumes its downstream node has been "dead" and starts to look for an alternative path that bypasses the "dead" node (named ***escaped node*** in the context) in its neighborhood. In this section, we will describe in details how to seek an alternative path. We will also discuss some issues involved and then refine the basic path repairing approach with three enhancements.

A. Alternate Path Selection Rule

Seeking an alternative path starts by the immediate upstream node of the broken link (called ***helped node*** or ***pivot node***) broadcasting a *Help Request* (*HREQ*) message. This message carries the following fields in its header: the helped node Id ID_{helped_node}, the escaped node's Id $ID_{escaped\_node}$, the escaped node's cost $C_{escaped\_node}$, the identification of the served data packet (*Seq_#* and *source_id*), as well as transmitter Id *sender_id*.

After receiving this message, each alive neighbor node performs the following comparisons sequentially based on the information previously stored and does the corresponding process according to the comparison result.

a) If the downstream node of the receiver, denoted as *j*, is the helped node or the escaped node (i.e., $HREQ.ID_{helped\_node} = routing\_table.ID_{downstream}$ or $HREQ.ID_{escaped\_node} = routing\_table.ID_{downstream}$), the message is discarded, because the purpose of alternative path seeking is to bypass the escaped node.

b) If the data packet, identified by *HREQ. Seq_# and HREQ.source_id*, has been received by the current node *j* before, the message is discarded, because previous receipt implies that the same packet had traversed the current node *j* and tried in vain to move along further.

c) If the comparison results are not the above two cases, the current node *j* compares its cost with that of the escaped node. If its cost is equal to or smaller *than HREQ.* $C_{escaped\_node}$, the current node believes that its optimal path to the destination will not go through the escaped node. Therefore, it issues a *Help Response* (*HREP*) message and sends it back to the helped node along the reverse path traveled by the *Help Request* message. Otherwise, the current node *j* passes the *Help Request* message on to its downstream node and that node will do the same comparisons and process as described in a-c).

Besides *Seq_#* and *source_id*, a *Help Response* message also carries another two fields in its header: transmitter Id *sender_id* and its cost *sender_cost*. Each *HREP* message received by the helped node indicates a qualified alternative path candidate from the helped node to the destination, which can bypass the escaped node. The cost of the alternative path candidate is *HREP. sender_cost +1* and the next immediate hop is *HREP. sender_id*. Thus the helped node will select the candidate with minimal cost as the alternative path, update its routing table accordingly, and then forward the data packet to its new downstream node.

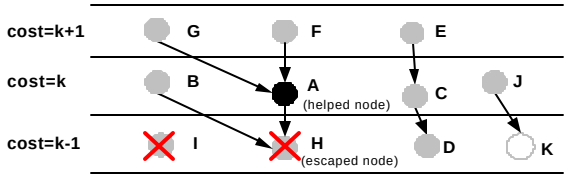

Figure 1: Alternate downstream node selection

An example of the above procedure is illustrated in Fig 1. Node *A* receives a data packet from its upstream node *F* and forwards the packet to its downstream node *H* that is dead. After detecting the transmission break, node *A* broadcasts a *Help Request* message to its neighbors, which are marked as gray solid circles in the figure. The costs of node *A* and its neighbors are labeled on the left. Each node has a black arrow going out and pointing to its downstream node recorded in its routing table. After receiving the *Help Request* message from Node *A*, node *B* discards it, because its downstream node is the escaped node *H*. Node *G* and node *F* discard the message as well, because the downstream nodes of node *G* and node *F* are both the helped node *A*.

Node *E* forwards the message to node *C*, because node *E*'s cost (= *k+1*) is greater than that of the escaped node *H* (= *k-1*). After receiving the *Help Request* message from node *E*, node *C* passes the message on to its downstream node. The process stops at node *D* whose cost is equal to that of node *H*. Then, node *D* issues a corresponding *Help Response* message and sends it back to node *A* along the path *D-C-E-A*. From this Help Response message, node *A* is aware of an alternative path candidate *A-E-…*(*A-E-…* stands for the path from node *A* to node *E* and then following node E's optimal path to the given sink node), whose cost is *k+2* (= 1+node *E*'s cost). Likewise, node *A* also receives *Help Response* messages from node *C*, *D* and *J* respectively. Among all the alternative path candidates, the path *A-D-…*has the lowest cost (=*k*). So node *A* updates its routing table by changing its downstream node Id $ID_{downstream}$ to *D*, its downstream node cost $C_{downstream}$ to *k-1* and the cost of itself *C* to *k*. After that, the data forwarding is resumed from node *A* along the new route.

B. Returning Packet If No Alternative Path Found

If no *Help Response* message has been received, the helped node sets the *direction* bit in the header of the data packet as 1 and returns the packet to the node from which it received the packet on the first time. After receiving the returned packet, that node initiates a *Help Request-Response* process in its local area and attempts to find an alternative path there. The whole data delivery stops when the data packet has reached the destination successfully or when the data packet has been retraced back to the source node and no alternative path is found any more.

C. Postponing Cost Convergence

After path repairs, the costs stored in routing tables of some nodes may not be up to date any more. For instance, as shown in Fig 1, supposing node *D* is also dead and Node *A* has to select node *J* as its new downstream node. Thus, the cost of node *A* is updated from *k* to *k+1*. All the nodes along the upstream branches of node *A* (including node *G* and node *F*) should be updated correspondingly. However, in order to avoid the communication overhead from using dedicated messages for immediate update, our algorithm uses a postponed cost convergence scheme: every backward message (including returned data packet and *Help Response* message) piggybacks the latest cost value of the transmitter in its *sender_cost* field. The receiver changes its downstream node cost $C_{downstream}$ to *sender_cost* and its cost value *C* to *sender_cost*+1 after receiving the piggybacked information. In other words, alternation of cost at one node isn't notified to its upstream nodes until backward messages destined to them need to be transmitted.

D. Enhancement

1) Loop Prevention

The above postponed cost convergence scheme has no extra transmission and energy cost, however, the "staleness" of cost values may cause loops in both the data forwarding process and *Help-Request-Response* process. To prevent loops, we add two control schemes in our algorithm:

a) If a node receives a forwarded data packet and it finds that it has received the same packet before, it immediately initiates a *Help-Request-Response* process to seek an alternative path bypassing its downstream node, instead of forwarding the packet to that node.

b) A *TTL* field is added to *Help Request* message to limit the length of the route the message can travel. In our preliminary simulation, we set the *TTL* field to 3. How different *TTL* values affect the performance of our algorithm is one part of our future work.

2) Limit Repair Number

During the simulation, we found that in some network topologies, especially those with a low density and a high node failure ratio, failure of some critical nodes may cause the path unable to be repaired. Unrestricted repair cannot improve the delivery ratio, while wasting energy substantially. To solve this problem, we introduce two fields in the header of a data packet, *repair_num* and *repair_limit*. *repair_num* is initially set to 0 and incremented by 1 when a *Help Request-Response* process is launched. *repair_limit* is used to control how many repairs are allowed for a data delivery and is set by the source node as a linear function of its cost ($=\alpha \times C$). When *repair_num* exceeds *repair_limit*, the data delivery terminates immediately. Statistical results show that when α is set to 3, the delivery ratio drops by less than 1% on average, while the energy consumption is reduced by more than 8%, even up to 80% in some cases. Therefore, we set α to 3 in our preliminary experiment.

3) Relax Rule c) In Alternative Path Selection

Rule c) in section III.A states that the qualification of an alternative path candidate cannot be confirmed until the *Help Request* message reaches the node, whose cost *C* equals to or lower than $HREQ.C_{escaped\_node}$. Such condition can be relaxed to $C_{downstream} <= HREQ.C_{escaped\_node}$. For instance, in Fig 1, the qualification of the alternative path candidate *A-E-C-D-…* is confirmed by node *C* other than node *D* and the qualification of the alternative path candidate *A-J-K-…* is confirmed by node *J* other than node *K*. Note that this relaxation conforms to the purpose of *Help Request-Response* process: to bypass the escaped node. It is not demanded that all the nodes in the alternative path must be alive. Furthermore, after the relaxation, energy can be conserved because a *Help Request* message travels in a shorter distance. Also, the data delivery ratio may be increased because the number of alternative path candidates is increased and the range of path repair is extended. In the example illustrated in Fig 1, assuming node *D* and node *J* are both dead, according to the new rule c), paths *E-C-D-…* and *C-D-…* are both qualified alternative path candidates (according to the old one, they aren't). Therefore, node *A* selects node *C* as its new downstream node. After the data packet reaches node *C*, another repair is launched by node *C* in its local area. This repair may be critical to the success of data delivery in some scenarios. We compared the difference of delivery ratio and energy consumption between these two rules in the test cases described in the next section and found that the delivery ratio is about 1% higher and energy consumption is 2% lower on

average by using the new rule c) than by using the original one.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the effectiveness of the proposed scheme through experiments. We choose two metrics to investigate the performance of our scheme and compare it to other schemes: delivery ratio and average energy consumption per data delivery. The delivery ratio is defined as the ratio of the number of data packets successfully received by a sink node to the total number of data packets sent by source nodes. The average energy consumption is the ratio of the total energy dissipation to the total number of delivered data packets. For the proposed scheme, the energy consumed in path repairs is also included. We study these metrics as a function of node density, source-sink distance, node failure ratio and channel error rate.

The routing schemes we evaluated are the following:

a) Data forwarding along single path with repair (SWR): the proposed scheme

b) Data forwarding along single path without repair (SWOR): Data packet sent from the source node is forwarded along the pre-determined lowest-cost path until a broken link is encountered or it reaches the destination. No repair is performed against the broken link.

c) Data forwarding along 2-Disjointed paths without repair (DISJ2): Two copies of the data packet are delivered separately along two pre-determined paths. The first path is the lowest cost path. The second is the best path among all the paths that are node-disjointed from the first one.

d) Mesh data forwarding (MESH_1.3, MESH_0.13): This scheme is proposed in [5]. The basic idea is to let each data packet carry a credit, which specifies the amount of extra cost allowed beyond the source node's minimum cost to the sink node. Multiple copies of a data packet can go along different paths that interweave and form a mesh. The credit is set as $\alpha \times C_{source}$, in which the value of α determines the width of the mesh and thus the delivery ratio and energy consumption. Larger α results in a wider mesh and in turn a higher delivery ratio and more energy consumption. In our simulation, α is set to 1.3 and 0.13 separately to represent wide and narrow mesh forwarding schemes.

Experiments are carried out using random network topologies. In each network topology, n nodes are randomly scattered in a fixed 100×100 m^2 deployed area. The sink node is located at (100,100)-the lower right corner of the deployed area. Each node has a maximum transmission range of 10 meters. Before data delivery, the routing table has been set up at each sensor node. The energy dissipated in the optimal path establishment process is not included in the total energy consumption, because all the schemes listed above need the same process. *DISJ2* requires extra overhead to set up the second path that is node-disjointed from the first one. We don't include this part into its total energy consumption, either.

For each network topology, source nodes are those nodes, whose optimal path length - cost value is L. Each source node sends one data packet to the sink node, using the above schemes respectively. After routing table establishment and before data delivery, failure nodes are selected randomly from all the sensor nodes. The ratio of failure node number to the total node number is P_n. To make fair comparisons, all schemes use exactly the same network topologies and the same failure nodes in each network topology. We also introduce a channel error rate P_l in each one-hop transmission. In order to study the performance of the different schemes as a function of node density, we keep other parameters constant (L=10-hop and P_l=P_n=10%) and change n from 200 to 600 with increments of 50. Similarly, we vary L from 4 hops to 16 hops with increments of 2 hops, while fixing n=400 and P_l=P_n=10%, to get their performance curves as a function of the source-sink distance. We also change P_n or P_l from 0% to 30% with n=400, L=10-hop and P_l or P_n =10% to get their behaviors as a function of node failure ratio or channel error rate respectively.

We use a simple abstract energy model other than a particular one to compute energy consumption in one data delivery. This model assumes that the power required for receiving one control packet (*HREQ* or *HREP* message) is 1 energy unit. If the energy for transmitting one bit is t times as much as the energy for receiving one bit, the power required for transmitting one control packet is t energy units. Let the ratio of the data packet size to the control packet size be s, the power required for receiving one data packet is s energy units and that for transmitting one data packet is $s \times t$ energy units. Such abstract energy model is justified, for we are interested in comparison results and change tendencies of all the schemes, rather than exactly real values. In the preliminary experiment, we set s to 5 and set t to 1. Investigating the performance under other s and t values is one part of our future work.

Fig. 2 shows delivery ratio and energy consumption as a function of node density n. We also present the ratio of energy consumption to delivery ratio as a metric when considering both of them at the same time. The smaller the cost to success ratio is, the more efficient the scheme is. We have the following main observations from Fig. 2 : 1). The delivery ratios of *SWOR* and *DISJ2* almost fluctuate at 16% and 26% respectively, and doesn't have obvious rise or fall with the change of n. In contrast, the delivery ratios of *SWR* and *MESH* grow as n increases, which is because path reparability is improved or mesh width is broadened at higher node density. When n reaches 550, the delivery ratios of *SWR* and *MESH_1.3* are both higher than 95%. Overall, the delivery ratio of *SWR* is the highest among all the compared schemes. 2). The energy consumptions in all the schemes grow with n. The main reason is that more power is dissipated for overhearing when every node has more

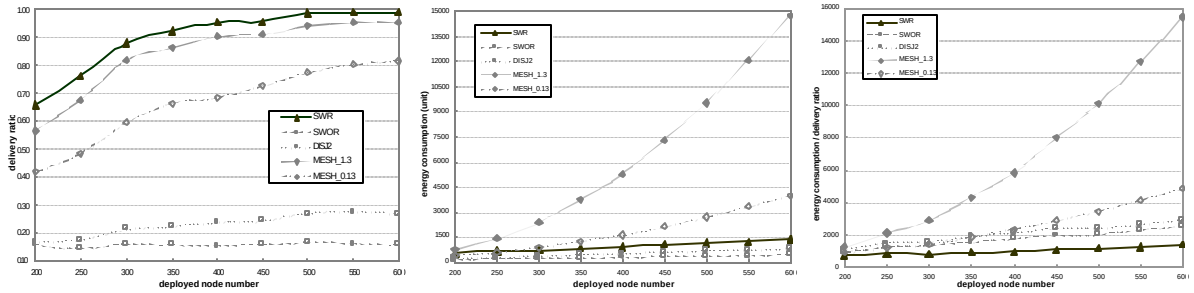

Figure 2: delivery ratio and energy consumption vs. node density (10-hop source-sink distance, 10% node failure ratio, 10% channel error rate)

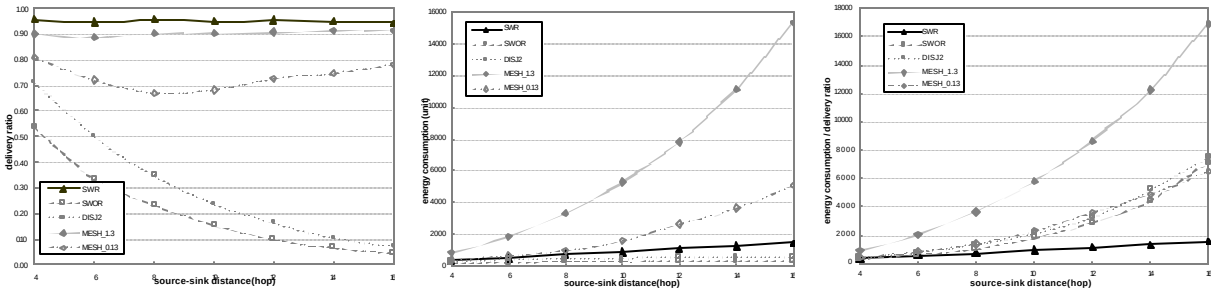

Figure 3: delivery ratio and energy consumption vs. path length (400 deployed nodes, 10% node failure ratio, 10% channel error rate)

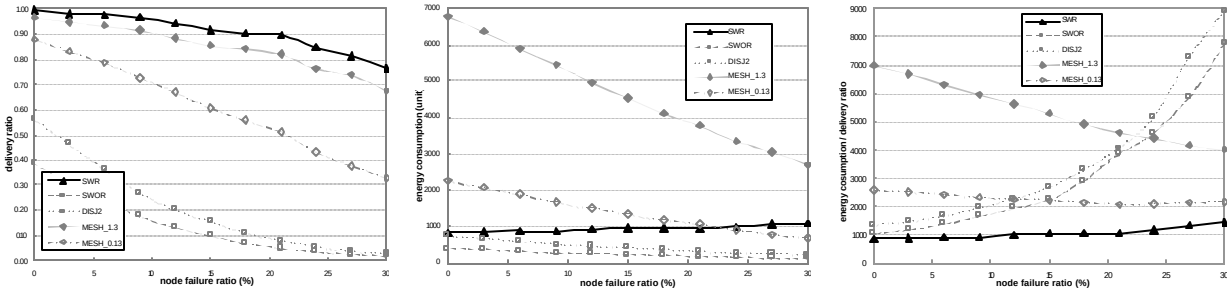

Figure 4: delivery ratio and energy consumption vs. node failure ratio (400 deployed nodes, 10-hop source-sink distance, 10% channel error rate)

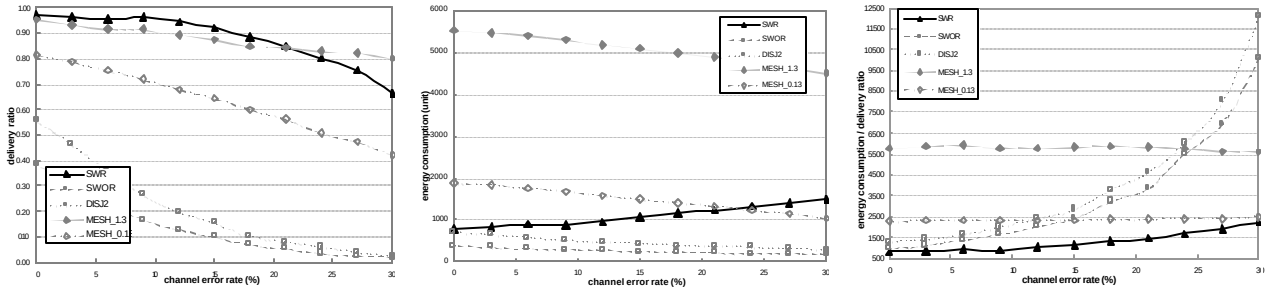

Figure 5: delivery ratio and energy consumption vs. channel error rate (400 deployed nodes, 10-hop source-sink distance, 10% node failure ratio)

neighbors. However, compared with the other schemes, the increase in *MESH* is quick and dramatic. E.g. when n is 200, the energy units consumed by *MESH_1.3* and *MESH_0.13* are 0.47% more and 0.25% less than those by *SWR* respectively. While when n is 600, the energy units consumed by *MESH_1.3* and *MESH_0.13* are about 11 times and 3 times as many as those by *SWR* respectively. Such dramatic growth of energy consumption in *MESH* results from the fact that the mesh width in *MESH* is inherently broadened with the node density.

Fig. 3 plots the change of delivery ratio and energy consumption vs. different source-sink distances L. Delivery ratios in *SWR* and *MESH* schemes have no distinct changes as L increases. *SWR* maintains 95% success ratio even when L=16 hops. In contrast, the delivery ratios in *SWOR* and *DISJ2* decrease dramatically with the increase of L. When L =16 hops, the delivery ratios of *SWOR* and *DISJ2* are both lower than 10%. Overall, the delivery ratio of *SWR* is the highest among all the compared schemes. Furthermore, the energy consumed by all these schemes increases as L increases. However, the increase in *SWR* is much slowly than

that in *MESH*. When L=16 hops, the energy units consumed by *MESH_0.13* and *MESH_1.3* are about 10 times and 4 times as many as those by *SWR* respectively.

Fig. 4 shows the change of delivery ratio and energy consumption as a function of node failure ratios P_n. As shown in the figure, the delivery ratios of all the schemes decrease as P_n increases. However, the decrease in *SWR* is much slower than those in *SWOR*, *DISJ2*, *MESH_0.13* and slightly slower than that in *MESH_1.3*. When P_n is as high as 30%, *SWR* still keeps the highest delivery ratio. Furthermore, at low node failure ratios, *MESH*s have a significant energy waste compared with the others. Except in *SWR*, energy consumptions in the other schemes decrease as P_n increases. That is because only *SWR* takes an active strategy against broken paths and always tries to repair them, which are at the cost of energy consequentially. In spite of this, when P_n is 30%, the energy units consumed by *SWR* are still 60% less than those by *MESH_1.3*, which the only one among all the schemes having a comparable delivery ratio with *SWR*.

Fig. 5 shows the change of delivery ratio and energy consumption as a function of channel error rate P_l. As shown in the figure, the delivery ratios of all the schemes decrease as P_l increases. However, the decrease in *SWR* becomes faster than that in *MESH_1.3* when P_l reaches 18%. When P_l is higher than 21%, the delivery ratio of *SWR* is lower than that of *MESH_1.3*. The reason is there is no mechanism in *SWR* to prevent lost of a retumed data packet, which terminates the whole data delivery and its possibility increases with P_l. In spite of this, in term of cost to success ratio, *SWR* is still more efficient than *MESH_1.3* even at 30% P_l. Improving the delivery ratio of *SWR* at a high channel error rate is one part of our future work.

V. CONCLUSION

This paper focuses on reliable data delivery schemes in wireless sensor networks. We believe that a single-path routing with repair is an effecctive solution, which can simultaneously guarantee delivery and avoid energy waste. We propose a path repairing approach, which can quickly find an alternate path against a broken link by doing a small survey around the break and only using already existing routing information. The proposed scheme *SWR* is implemented and compared with three other routing schemes (*SWOR*, *DISJ2* and *MESH*). Our experimental results show that *SWR* can provide the highest success ratio among all the compared schemes in a wide range of node density, path length, node failure ratio. It can also guarantee the highest success ratio at low and medium channel error rate. Furthermore, the energy consumed by *SWR* is lower than *MESH_1.3*, which is the only one having a comparable delivery ratio with *SWR* among all the evaluated schemes. Its advantage of energy efficiency is more remarkable at higher node density, longer source-sink distance and low path failure ratio.

In future, we will evaluate the impact of different parameter setting to the comparative results. We will improve the delivery ratio of the proposed scheme at a high channel error rate. Also, we will intend to investigate the worst case performance of the proposed scheme.

REFERENCES

[1] D. Estrin, R. Govindan, J. Heidemann and S. Kumar. "Next Century Challenges: Scalable Coordination in Sensor Networks". *ACM MobiCom'99* Washington 1999: 263-270

[2] J. Kahn, R. Katz, and K. Pister. "Next Century Challenges: Mobile Networking for Smart Dust". *ACM MobiCom'99* Washington 1999; 271-278

[3] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton and J.Zhao. "Habitat Monitoring: Application Driver for Wireless Communications Technology." *ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean* Costa Rica 2001; 3-5

[4] I. F. Akyildiz, W. Su, Y.Sankarasubramaniam, E.Cayirci. "Wireless Sensor Networks: A Survey." *Computer Networks*, March 2002

[5] F.Ye, S. Lu, and L.Zhang. Gradient Broadcast: "A Robust, Long-lived Sensor Network." Http://irl.cs.ucla.edu/papers/grab-tech-reports.ps, 2001

[6] 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.

[7] Y.Yu, R.Govindan, D.Estrin. "Geographical and energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks." Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Dept., May 2001.

[8] C. Intagagonwiwat, R.Govindan, and D.Estrin. "Directed Diffusion: A Salable and Robust Communication Paradigm for Sensor Networks." *Mobicom'00* Boston 2000; 56-67

[9] R.C.Shah, H.M.Rabaey. "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks." *IEEE Wireless Communications and Networking Conference*, 2002

[10] B.Krishnamachari, D.Estrin, S.Wicker. "Modelling Data-Centric Routing in Wireless Sensor Networks." *IEEE INFOCOM* 2002

[11] W. R. Heizelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for wireless Micro sensor Networks," IEEE Proceedings of the Hawaii International Conference on System Sciences, pp.-10, January 2000.

[12] D.Braginsky, D.Estrin. "Rumor Routing Algorithm for sensor Networks." In Proc. of the First ACM International workshop on Wireless Sensor Networks &Applications, Atlanta, September 2002, page 22-29

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

[14] D.B.Johnson, D.A.Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks. Mobile computing," T.Imielinske and H.Korth, eds., Kluwer Academic Publishers, Norwell, Mass., 1996, 153-181.

[15] C.Y.Wan, A.T.Campbell, L.Krishnamurthy. "PSFQ: A Reliable Transport Protocol for wireless Sensor Networks." In Proc. of the First ACM International workshop on Wireless Sensor Networks &Applications, Atlanta, September 2002, page 1-11