Energy Efficient Opportunistic Routing in Wireless Sensor Networks

Xufei Mao, *Member, IEEE,* Shaojie Tang, *Student Member, IEEE,* Xiahua Xu, *Student Member, IEEE,* Xiang-Yang Li, *Senior Member, IEEE,* Huadong Ma, *Member, IEEE,*

Abstract—Opportunistic routing [2], [3] has been shown to improve the network throughput, by allowing nodes that overhear the transmission and closer to the destination to participate in forwarding the packet, *i.e.*, in *forwarder list*. The nodes in *forwarder list* are prioritized and the lower priority forwarder will discard the packet if the packet has been forwarded by a higher priority forwarder. One challenging problem is to select and prioritize forwarder list such that a certain network performance is optimized. In this paper, we focus on selecting and prioritizing forwarder list to minimize energy consumptions by all nodes. We study both cases where the transmission power of each node is fixed or dynamically adjustable. We present an energy efficient opportunistic routing strategy, denoted as EEOR. Our extensive simulations in TOSSIM show that our protocol EEOR performs better than the well-known ExOR protocol (when adapted in sensor networks) in terms of the energy consumption, the packet loss ratio, the average delivery delay.

Index Terms—Sensor networks, opportunistic routing, energy.

1 Introduction

Routing protocol design for wireless networks are often guided by two essential requirements: minimize energy cost or maximize network throughput. The traditional routing protocols in wired networks choose the best sequence of nodes between the source and destination, and forward each packet through that sequence. The majority routing protocols designed for multi-hop wireless networks have typically followed this convention, including those multi-path routing protocols. However, this did not take advantages of the broadcast nature of wireless communications: a node's transmission could be heard by any node within its transmission range. On the other hand, the lossy and dynamic wireless links make it difficult for traditional routing protocols to achieve stable performances.

In wireless networks, various factors, like fading, interference, and multi-path effects, can lead to temporary heavy packet losses [11] in a pre-selected *good* path. In contrast, opportunistic routing, like ExOR [2] and MORE [3], allows any node that overhears the transmission to participate in forwarding the packet. The routing path is selected on the fly and completely opportunistic based on the current link quality situations. However, this new design paradigm introduces several challenges. One challenge is that multiple nodes may hear a packet and unnecessarily forward the same packet. ExOR deals with this challenge by tying the MAC to the routing, imposing a strict scheduler on routers access to the medium. The scheduler goes in rounds. Forwarders transmit in order such that only one forwarder is allowed to transmit at any time. The other forwarders listen to the transmissions to learn which packets were overheard by each node. In contrast to ExOR's highly structured scheduler, MORE [3] addresses this challenge with randomness. MORE randomly mixes packets before forwarding them. This ensures that routers which hear the same transmission do not forward the same packet. As a result, MORE does not need a special scheduler; it runs directly on top of 802.11. Both ExOR and MORE showed that this kind of opportunistic routing strategy can improve the wireless network's performance.

In this paper, we study how to select and prioritize the forwarding list to minimize the total energy cost of forwarding data to the sink node in a wireless sensor network. Observe that previous protocols, *i.e.*, ExOR and MORE, did not explore the benefit of selecting the appropriate forwarding list to minimize the energy cost. We will investigate this problem through rigorous theoretical analysis as well as extensive simulations. We study two complementary cases (1) the transmission power of each node is fixed (known as *non-adjustable transmission model*) and (2) each node can adjust its transmission power for each transmission (known as *adjustable transmission model*). Optimum algorithms to select and prioritize forwarder list in both cases are presented and analyzed. It is worth to mention that our analysis does not assume any special energy models. We conducted extensive simulations in TOSSIM to study the performance of proposed algorithms by comparing it with ExOR [2] and traditional routing protocols. It shows that the energy consumption of routing using EEOR is significantly lower than ExOR with random forwarder list and traditional distance vector routing protocols.

- *The research of authors is partially supported by NSF CNS-0832120, NSF CNS-1035894, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program), the National Basic Research Program of China (973 Program) under Grant No.2011CB302701, the National Natural Science Foundation of China under Grant No.60833009, the National Science Funds for Distinguished Young Scientists under Grant No.60925010.*
- *Mao and Ma are with Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing China. Tang, Xu, and Li are with Department of Computer Science, Illinois Institute of Technology, Chicago. Emails: maoxufei@bupt.edu.cn, stang7@iit.edu, xxu23@iit.edu, xli@cs.iit.edu, mhd@bupt.edu.cn*

2 Network Model and Preliminary

We consider a wireless sensor network and assume that all wireless nodes have distinctive identities, *i.e.*, $i \in [1, n]$. In Section 3 we first assume that every wireless node u has fixed transmission power W. In Section 4, we assume that each node can adjust its transmission power to any value between 0 and W. Let w denote such adjusted transmission power. The multihop wireless network is then modeled by a communication graph $G = (V, E)$, where V is a set of $n = |V|$ wireless nodes and E is a set of directed links. Each directed link (u, v) has a non-negative *weight*, denoted by $w(u, v)$, which is the minimum transmission power required by node u to send a packet to node v successfully. It is worth to mention that our methods work with any weight function $w()$.

Since the number of neighboring nodes of a node u may change when different transmission power is used, we define $N_w(u)$ as the neighboring nodes of a node u when u transmits with the power w. For simplicity, when the subscript w is not mentioned, we mean that the node is using its maximum transmission power, i.e., $N(u) = N_W(u)$. In addition, each link (u, v) has an *error probability*, denoted by $e(u, v)$, which is the probability that a transmission over link (u, v) is not successful, i.e., node u must consume at least $w(u, v)$ power to have a chance of $1 - e(u, v)$ to transmit a packet to node v. No transmission is possible if less power is used.

To illustrate how we can take advantage of wireless broadcast advantage (WBA), let us consider a network example in Figure 1 (a). The error probability from the source node to each node v_i is e and the error probability from each node v_i to the target node is 0. Traditional routing would route all data packets through the same node, say v_i. The expected number of transmissions will be $\frac{1}{1-e}$ for the intended node v_i to receive the packet correctly. On the other hand, by taking advantage of WBA property, by letting every intermediate node v_j to listen to the transmissions, the expected number of transmissions is reduced to $\frac{1}{1-e^n}$ for at least one node to receive the packet correctly. This difference will be more noticeable when e is close to 1 and n is a big number.

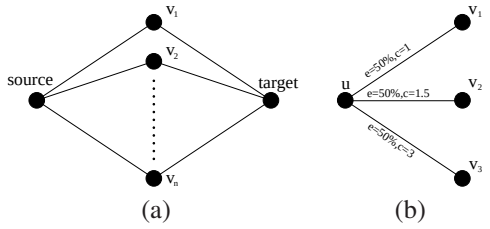


Fig. 1. (a) Wireless Broadcast Advantage. (b) Calculating the expected cost.

The advantage of WBA is more obvious in a multi-hop wireless network, especially when a source node and the destination node are far way, i.e., the packet from the source node to a target node must be routed through a multi-hop path. As proposed in ExOR [2], the source node selects a subset of its neighboring nodes as forwarder list. The forwarder list is prioritized to indicate which nodes have higher priority to forward the packet. Then one or more nodes in the forwarder list, which received the packet successfully, will opportunistically act as new source nodes and route the packet to the target node.

In summary, the main idea of opportunistic routing are as follows. We let $C_u(\mathrm{Fwd})$ denote the expected cost needed by the node u using opportunistic routing strategy to send a packet to the target node when the forwarder list chosen by u is Fwd. For simplicity, we use C_u to denote the expected cost of node u if there are no confusions. Initially, the expected cost of the target node is set to be 0 and the costs of all other nodes are set to be ∞. Using the similar mechanism of distance vector routing, the calculations of the expect cost for each node will be carried out periodically and every node updates its expected cost and forwarder list periodically. When a node needs to send or relay a packet to some destination node, it will simply broadcast the packet and let some node(s) in its forwarder list (constructed according to the destination node) to recursively forward the data packet. In the next two sections, we will focus on how to compute the expect cost and choose the forwarder list for each wireless node: Section 3 focuses on the fixed transmission power case and Section 4 focuses on the case when nodes can dynamically adjust the transmission power.

3 Non-Adjustable Power Model

We consider the case when each node uses a fixed transmission power. One may think that the best forwarder list for a node u in this case is $N(u)$. Surprisingly, this is not always true. At the end of this section, we will show an example, based on the Figure 1, that the best forwarder list may be a subset of $N(u)$.

3.1 Compute the expected cost

Now we present the main idea on calculating the expected cost for each node and selecting the forwarder list. Consider a node u and its neighbors. We will compute the expected cost of and the forwarder list of node u based on the expected cost of its neighbors whose expected cost of sending data to the given target node has already been computed. In other words, here we want to choose a subset of neighboring nodes $N(u)$ as forwarder list of node u such that the expected cost for u to send a packet to the target is minimized. To understand our method better, we introduce some definitions first. Consider a fixed target. Given a set of nodes S, let S^* denote the increasingly sorted list of S based on the expected cost by each node in S to send data (via possible relay) to this given target node. Let $\mathrm{Fwd}(u)$ denote the forwarder list of node u.

To find the expected cost at node u, we first sort the forwarder list $\mathrm{Fwd}^*(u)$ in increasing order by the expected cost, *i.e.*, $\mathrm{Fwd}^*(u) = \{v_1, v_2, ..., v_{|\mathrm{Fwd}(u)|}\}$, where $i < j \Rightarrow C_{v_i} \leq C_{v_j}$. Let α denote the probability that a packet sent by node u is not received by any node in $\mathrm{Fwd}^*(u)$. Clearly,

$$\alpha = \prod_{i=1}^{|\mathrm{Fwd}^*(u)|} e_{uv_i} \quad (1)$$

Let ρ denote the probability that a packet sent by node u is received by at least one node in $\mathrm{Fwd}^*(u)$. Then $\rho = 1 - \alpha$.

Let $C_u^h(\mathrm{Fwd}^*)$ denote the expected energy that node u must consume to send a packet to at least one node in the forwarder list Fwd^*. $C_u^h(\mathrm{Fwd}^*)$ can be calculated as follows:

$$C_u^h(\mathrm{Fwd}^*) = \frac{w}{\rho} \tag{2}$$

When at least one node in the forwarder list received the packet successfully, we need to calculate the expected cost to forward the packet sent by node u. Here we assume that only one node from the forwarder list that received the packet will forward the packet. Although this assumption is very optimistic, as we will explain later, in most cases it is true. The expected cost that we calculate here could be slightly lower than the actual cost when multiple nodes from forwarder list could forward the data packet.

Let $C_u^f(\mathrm{Fwd}^*)$ denote the expected total cost for u to forward (using some nodes in the forwarder list of u) the packet to the target. $C_u^f(\mathrm{Fwd}^*)$ can be calculated as follows. Assume the prioritized forwarder list is $\mathrm{Fwd}^* = \{v_1, v_2, \cdots, v_{|\mathrm{Fwd}^*|}\}$. The probability that node v_1 forwards the packet is $1-e(u,v_1)$ and the expected cost by v_1 is C_{v_1}; then node v_2 will forward the packet with probability $e(u,v_1)\cdot(1-e(u,v_2))$ and the cost will be C_{v_2}. Basically, node v_i forwards the packet if it receives the packet and nodes $v_j, 0<j<i$ did not receive the packet, and in this case, the cost will be C_{v_j}. Hence, the expected cost can be computed as follows:

$$\beta = (1-e_{uv_1})C_{v_1} + \sum_{i=2}^{|\mathrm{Fwd}^*|}\left(\prod_{j=1}^{i-1} e_{uv_j}\right)\cdot(1-e_{uv_i})\cdot C_{v_i} \tag{3}$$

Since β is computed under condition that a forwarder node got the packet, then we have

$$C_u^f(\mathrm{Fwd}^*) = \frac{\beta}{\rho} \tag{4}$$

Notice that the communication cost for obtaining agreement among nodes in Fwd on which node will forward data is also a factor that affects the total cost forwarding data in practice. Let $C_u^c(\mathrm{Fwd}^*)$ denote the communication cost from all nodes in the forwarder list in order to reach an agreement on which node will finally help to relay the packet, $C_u(\mathrm{Fwd}^*)$ is computed as follows:

$$C_u(\mathrm{Fwd}^*) = C_u^h(\mathrm{Fwd}^*) + C_u^f(\mathrm{Fwd}^*) + C_u^c(\mathrm{Fwd}^*) \tag{5}$$

Equation 5 illustrated how to compute the expected cost of a sender to broadcast a packet if the current chosen forwarder list is Fwd^*. The cost consists of three parts. The first part is the expected cost for the sender to successfully transmit a packet to at least one receiver in Fwd^*. The second part is the expected cost that there is one node in the forwarder list Fwd^* to help to relay the packet to the final destination node. The third part $C_u^c(\mathrm{Fwd}^*)$ is the communication cost to reach an agreement on choosing the actual relay node. This cost $C_u^c(\mathrm{Fwd}^*)$ is often incurred once when the network is static, while the cost of sending and forwarding depends on the traffic flows.

Without Agreement to Resolve Duplication: Observe that in our previous computation, we assume that we would like to pay an additional cost $C_u^c(\mathrm{Fwd}^*)$ among the forwarding nodes to prevent the scenario when multiple forwarding nodes receive the packet correctly and all decide to forward the packet. When this additional communication is not applied, potentially few nodes may forward the data. This happens when some receiving nodes in Fwd cannot hear from each other directly. Figure 2 illustrates such an example.

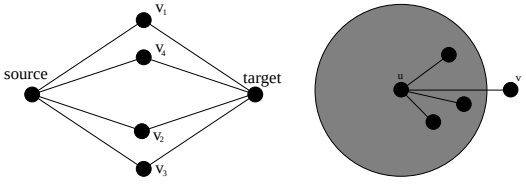


Fig. 2. (a) An example for expected cost calculation, (b) Calculating the expected cost in adjustable transmission power model.

In Fig 2, assume $\langle v_1, v_4\rangle$ and $\langle v_2, v_3\rangle$ are the only neighboring pairs among the forwarding list. If no communications are used to resolve duplicates, (*i.e.*, $C_u^c(\mathrm{Fwd}^*)=0$) then the forwarding cost can be calculated as

$$C_u^f(\mathrm{Fwd}^*) = \rho^{-1}\cdot((1-e_{uv_1})\cdot C_{v_1} + (1-e_{uv_2})\cdot C_{v_2} + e_{uv_2}\cdot(1-e_{uv_3})\cdot C_{v_3} + e_{uv_1}\cdot(1-e_{uv_4})\cdot C_{v_4})$$

In other words, a node v_i will forward the packet only if v_i received the packet, and all its neighboring nodes with higher priority did not forward the packet. Thus, the cost of forwarding is computed as follows:

$$C_u^f(\mathrm{Fwd}^*) = \frac{\sum_{i=1}^{|\mathrm{Fwd}^*|}\left(\prod_{j=1,v_j\in N(v_i)}^{i-1} e_{uv_j}\right)\cdot(1-e_{uv_i})\cdot C_{v_i}}{\rho} \tag{6}$$

Due to the hardness to estimate the agreement cost and considering that most strategies need to pay the communication cost in order to guarantee the 100% data transmission success ratio, we omit the communication cost for agreement when we compute the forwarding list, *i.e.* formula (6) will be used instead. However, we do count the number of ACK messages used by each node for each packet and use this data as the communication cost in our TOSSIM simulations. We admit that this is may be not accurate enough and we will do further analysis in our future work.

3.2 Finding the optimal forwarder list

So far we have introduced the method to calculate the expected cost for a given node when the forwarder list is given. Next, we discuss how to choose the forwarder list. Consider there are k nodes in $N(u)$ for which an expected cost is already assigned, then there are (2^k-1) choices to select the forwarder list. Finding the expected cost pertaining to each forwarder list is not practical. Here we study the properties of the forwarder list and the expected cost and then we explain how to efficiently choose the optimal forwarder list.

To simplify our arguments, let us introduce a property known as *prefix*. A set X is called a prefix of an ordered set

Y if X are the set of first k elements of Y. So each set Y has $(|Y|+1)$ prefixes. Now consider node u and its neighboring nodes $N(u)$. Sort the nodes in $N(u)$ based on their expected cost in increasing order, and get $N^*(u) = \{v_1, v_2, ..., v_{|N(u)|}\}$ such that $|N(u)| \geq i > j > 0 \Rightarrow C_{v_i} > C_{v_j}$. First we show that the optimum forwarder list of node u is a prefix of $N^*(u)$.

Theorem 1: [1] The optimum forwarder list of node u must be a prefix of $N^*(u)$.

We further study the properties of forwarder list by introducing another two theorems. The first theorem, Theorem 2, shows that if a node, whose expected cost is less than the expected cost of a prefix forwarder list, is added to the forwarder list, then the expected cost of the newly created forwarder list will decrease while it will still be greater than the expected cost of the newly added node. The second theorem, Theorem 3, shows that if a node, whose expected cost is greater than the expected cost of a prefix forwarder list, is added to the forwarder list, then the expected cost of the newly created forwarder list will increase.

Theorem 2: [1] Consider a node u, a prefix forwarder list Fwd^*, and a node $v_k \in N(u) \setminus \mathrm{Fwd}^*$. If $C_{v_k} < C_u(\mathrm{Fwd}^*)$, then $C_{v_k} < C_u(\mathrm{Fwd}^* \bigcup \{v_k\}) < C_u(\mathrm{Fwd}^*)$

Theorem 2 proves that the expected cost of each node is higher than the expected cost of every node in its forwarder list. This property enables us to take a greedy approach in routing, which will be discussed later.

Theorem 3: [1] Consider a node u, a prefix forwarder list Fwd^*, and a node $v_k \in N(u) \setminus \mathrm{Fwd}^*$. If $C_{v_k} > C_u(\mathrm{Fwd}^*)$, then $C_u(\mathrm{Fwd}^* \bigcup \{v_k\}) > C_u(\mathrm{Fwd}^*)$.

Having these three properties, forwarder list can be selected easily. Algorithm 1 finds the optimum forwarder list and calculates the expected cost for a wireless node. Algorithm 1 works as follows. First it calculates $N^*(u)$ and then adds nodes in $N(u)$ to the forwarder list as long as the cost is decreasing. Once the cost starts to increase, it terminates. Based on Theorem 2, before we add a node to the forwarder list we know this operation will increase or decrease the cost. Note that based on the theorems we proved above, it is obvious that Algorithm 1 finds the optimum forwarder list.

Algorithm 1: ExpectedCostFixedPower$(u, N(u), C_u, \mathrm{Fwd})$

Input: the expected cost of all its neighboring nodes
Output: the cost C_u and forwarder list Fwd.

1: Set $C_u = \infty$, $\mathrm{Fwd} = \emptyset$.
2: Sort the neighboring nodes $N^*(u) = \{v_1, v_2, ..., v_{|N(u)|}\}$ based on its expected cost in increasing order.
3: **for** $(i = 1; i \leq |N(u)|; i = i + 1)$ **do**
4: **if** $(C_u > C_{v_i})$ **then**
5: Set $\mathrm{Fwd} = \mathrm{Fwd} \bigcup v_i$ and compute $C_u = C_u(\mathrm{Fwd})$ based on Equation (5).

Now we are ready to verify our claim that a node may not choose all its neighbors into the forwarder list as the optimum forwarder list at the beginning of this section. Consider a network example illustrated by Figure 1 (b). Assume node u consumes one unit of energy (*i.e.* $w = 1$) to send a packet and $N_1(u) = \{v_1, v_2, v_3\}$. For simplicity let e_i denote

Algorithm 2: ExpectedCostAdjustPower(u, C_u, Fwd)

1: Set $C_u = \infty$, $\mathrm{Fwd} = \emptyset$
2: Sort nodes in $N(u)$ based on *weight* in increasing order.
3: Let $N(u) = \{v_1, v_2, ..., v_{|N(u)|}\}$
4: **for** $(i = 1; i \leq |N(u)|; i = i + 1)$ **do**
5: Set $w = w(u, v_i)$
6: Run Algorithm 1, ExpectedCostFixedPower$(u, N_w(u), CrCost, \mathrm{CFwd})$
7: **if** $C_u > CrCost$ **then**
8: Set $C_u = CrCost$ and $\mathrm{Fwd} = \mathrm{CFwd}$.

$e(u, v_i)$ and c_i denote the expected cost at node v_i. It is desired to calculate the expected cost at node u. First we add node v_1 to the forwarder list. The expected cost if $\mathrm{Fwd}(u) = \{v_1\}$ will be $\frac{w+(1-e_1)\cdot c_1}{1-e_1} = 3$. The expected cost at node v_2 is 1.5, so based on Theorem 2 adding node v_2 will decrease the expected cost at node u. The expected cost if $\mathrm{Fwd}(u) = \{v_1, v_2\}$ will be $\frac{w+(1-e_1)\cdot c_1+e_1(1-e_2)\cdot c_2}{1-e_1e_2} = 2.5$. The expected at node v_3 is 3, so based on Theorem 3 adding node v_3 will increase the expected cost at node u. The expected cost if $\mathrm{Fwd}(u) = \{v_1, v_2, v_3\}$ will be $\frac{w+(1-e_1)\cdot c_1+e_1(1-e_2)\cdot c_2+e_1e_2(1-e_3)\cdot c_3}{1-e_1e_2e_3}$, which is equal to $\frac{18}{7} > 2.5$. So the optimum forwarder list is $\{v_1, v_2\}$ and the expected cost at node u is 2.5. This would serve as a good example that an optimum forwarder list is not necessarily $N(u)$, as mentioned in the beginning of this section.

4 Adjustable Power Model

In this section we consider the case where a node can adjust its power to any value $w \in [0, W]$. Note that for a given forwarder list, if we decrease w to the weight of the farthest link in $\mathrm{Fwd}(u)$ then C_u^h (see Equation 2) may decrease while C_u^f (see Equation 4) will remain the same, so using adjustable transmission ranges will give us some marginal improvement. As another example consider Figure 2. Assume node u has an expected cost of C_u when the transmission power w is used, where $W > w(u, v) > w$. As can be seen in Figure 2, if node u consumes power w, node v will not receive packets sent by node u. Should we increase the transmission power of node u to include node v in its transmission range? If $C_v > C_u$, based on Theorem 3, adding node v will increase the expected cost of node u even if no more additional power is needed. But if $C_v < C_u$, there is a tradeoff. On the one hand, adding node v increases the power C_u^h that node u must consume; on the other hand, decreases C_u^f may or may not decrease the expected cost at node u.

To find the expected cost in adjustable transmission power model, we sort the nodes in $N(u)$ based on the weight of the link that connects that node to u. Then we keep increasing the power at node u such that the number of nodes in $N_w(u)$ increases by one at each step until u reaches its transmission power limit or there is no more neighbor. Then for each w and each $N_w(u)$, using the Algorithm 1, we calculate the expected cost and pick the one that induces the minimum cost. Algorithm 2 summarizes our approach.

Next, we present our method (Algorithm 3) that builds the forwarder list for each node in the graph. We will calculate the *expected cost* for each node u to send a packet to the target node t. Let $C_{u,t}$ denote this expected cost and assume that the cost for a node to send a packet to itself is zero (*i.e.*, $C_{t,t} = 0$). Given a set V of nodes, a source node s, and a target node t, Algorithm 3 computes the expected energy cost needed to relay a packet from any node to the target node t using our opportunistic routing strategy.

Algorithm 3 works as follows. First, the set of nodes V is divided into two sets S_1 and S_2. Initially set $S_1 = V - \{t\}$ and $S_2 = \{t\}$. Then we find the node u in S_1 that has the least expected cost (denoted as $C_{u,t}$). We remove that node from S_1 and add it to Set S_2. The algorithm continues till all node are in the set S_2.

Let *Expected Cost Graph* denote the directed subgraph that includes a directed edge uv from the original communication graph if v is in the forwarder list of u. We have the following

Theorem 4: [1] Expected Cost Graph is loop free and Algorithm 3 assigns the optimum expected cost to each node.

Algorithm 3: Expected Cost by Opportunistic Routing

Input: target node t, source node s, power $w(u, v)$ and link reliability for each link uv.
Output: the expected cost $C_{u,t}$ from node u to node t using opportunistic routing and the forwarder list of each node u.

1: $\forall u \in V$, set $C_{u,t} = \infty$. Let $C_{t,t} = 0$.
2: $\forall u \in N(t)$ run Algorithm 1 or 2 to compute $C_{u,t} \Leftarrow C_u$.
3: **repeat**
4: Let v be the node in S_1 that has the minimum cost.
5: Let $S_1 = S_1 - \{v\}$ and $S_2 = S_2 \cup \{v\}$.
6: For each $u \in N(v) \cap S_1$, run Algorithm 1 or 2 to compute $C_{u,t}$, depending on the power model.
7: **until** no node updated the forwarder list and cost $C_{u,t}$.

Observe that the unmarked node u with the minimum cost among all unmarked nodes can be found using a distributed approach. However, the cost may be prohibitive. We thus design a method (Algorithm 4) that is similar to the Bellman-Ford algorithm, a distributed computing method of the shortest path. The basic idea of Algorithm 4 is to let each node continuously update its expected cost to the target node t. When the network does not change, the expected cost $C_{u,t}$ will not be reduced. The algorithm terminates when no node can reduce its expected cost $C_{u,t}$. It is easy to show that Algorithm 4 can terminate in constant rounds and find the correct optimum forwarder list and the cost $C_{u,t}$.

Algorithm 4: Distributed Computing of Forwarder List and Expected Cost by Opportunistic Routing

Input: target node t, source node s, power $w(u, v)$ and link reliability for each link uv.
Output: the expected cost $C_{u,t}$ from node u to node t using opportunistic routing and the forwarder list of each node u.

1: $\forall u \in V$, set $C_{u,t} = \infty$. Let $C_{t,t} = 0$.
2: $\forall u \in N(t)$ run Algorithm 1 or 2 to compute $C_{u,t} \Leftarrow C_u$.
3: **repeat**
4: For each u, run Algorithm 1 or 2 to compute $C_{u,t}$ and update its forwarder list, depending on the power model.
5: Node u sends the new cost $C_{u,t}$ to all its neighboring nodes.
6: **until** no node updated the forwarder list and cost $C_{u,t}$.

5 Performance Study in WSNs

In this section, we present the design details of our Energy Efficient Opportunistic Routing (EEOR) protocol in TinyOS-based wireless sensor network (WSN) simulation environment. In our simulation, we consider the case where there are multiple source/destination pair nodes in a randomly deployed WSN. Our design faces several key challenges. Firstly, all nodes in the forwarder list of a node must agree on next operation, *i.e.*, based on the priorities coming with the packet, which one(s) will finally act as the relay node(s) in order to save energy and increase the throughput. Since agreement involves communication and thus increases the overhead of the wireless network, we must guarantee the increased overhead will not overwhelm the performance gain brought by EEOR. Secondly, the EEOR protocol should be able to handle the network traffic efficiently, *i.e.*, be able to handle with congestion, to avoid bottleneck in order to decrease packet loss ratio and save the energy cost at the same time. To solve this issue, we need to consider many aspects. For example, the ongoing traffic flows from all source nodes should not exceed the capacity bound of the wireless networks. In other words, all source nodes should be able to dynamically adjust their network flows such that the ongoing flows in the wireless network are stable, e.g., push more flow to the network if the network does not reach its capacity; Otherwise, decrease its flow. Thirdly, a single packet could arrive at the destination through multiple pathes, thus involves more wireless nodes, consumes more energy and increases the traffic burden of wireless networks. Thus, it is necessary to introduce certain penalty scheme in order to punish those selfish nodes, e.g., some node chooses too many nodes as potential forwarders. This is because when a wireless node finds that the packets from its neighbor contain too many nodes in the forwarder list, it could increase its expected cost to quit the forwarder list next time or drop this packet. Fourthly, a node can utilize overheard messages to reduce the needs of ACK messages. Actually, to utilize these snooped information to avoid duplication is one important strategy in our design and simulation results indicate that this strategy can improve the system performance.

We implemented our protocol EEOR on TOSSIM, TinyOS 2.0.2. on Ubuntu 7.0.4. and conducted extensive tests based on different network environment. We compared our simulation results with ExOR [2] for unicast case in terms of energy consumption, packet loss ratio, end-to-end delay and packet duplication ratio. The experimental results showed that the performance of our protocol is better than ExOR's.

5.1 Network Description

We randomly place 100 wireless nodes with transmission range 50 feet in a 300×300 $feet^2$ square region. A node uses default CSMA MAC protocol in TinyOS.

From 100 wireless nodes we randomly pick 18 pairs of wireless node as source/destination pairs and for each source/destination pair nodes u and v, u will generate a new packet per second, which is heading for v by one- or multi-hop. Notice that the frequency of generating new packet could change when the source node find congestion in the network. We call the number of sending packets as *data size*. Considering the limited storage capacity of wireless sensor nodes, we set the buffer size to 20. After the buffer of a node is full, it will either drop new packet or replace old packet with new one according to different priorities of packets.

5.2 Performance Evaluations

We compared our protocol with ExOR with respect to the total energy consumption, packet loss rate, end-to-end delay and packet duplication ratio. We implement ExOR following the descriptions in [2]. To compare two protocols fairly, we use same max forwarder list size for both protocols and we let the each batch contain one packet in ExOR.

Due to different operations have different energy consumption parameters, we first considered and compared several operations of nodes which dominates the energy consumption, like sending and receiving. The Fig. 3 and 4 show the total transmission times and receiving times (including receiving, snooping intercepting) of all wireless nodes for both EEOR and ExOR.

As we can see from the figures, both transmission times and receiving times of ExOR are larger than EEOR's. This is due to the following reasons. First, for a node u in ExOR, it will always choose more neighbors (ExOR includes nodes that make on average at least 10% of the total expected number of transmissions [20]) into forwarder list for a packet under the constraint of penalty. However, in EEOR, when a node u chooses forwarder list for a packet, it will not only consider the expected cost of sorted neighbors, but also consider the increment cost by adding a node to the forwarder list such that u will not add a new neighbor to the forwarder list if doing so will increase the expected cost. Second, in ExOR protocol, a wireless node u's expected cost only depends on the neighbor which has smallest ETX value. However, the expected cost of a wireless node u is determined by the current selected forwarder list and link error rates between u and nodes in the forwarder list, which is more reasonable. These two differences between EEOR and ExOR make the average forwarder list size of the former is smaller than latter's in most of cases, thus EEOR involves fewer intermediate nodes.

Next, we measure the total energy consumption for both protocols based on the energy consumption parameters of TmoteSky sensor node. For example, the energy consumption for one time transmission and receiving for TmoteSky sensor node is $17.4mA$ and $19.7mA$ respectively. Given a fixed randomly topology, we randomly chosen 18 source/destination pairs, the Fig. 5 illustrates the total energy consumption for EEOR, AdjustablePower-EEOR and ExOR when we let the data size of each source node change from 200 to 500. As we can see, the total energy consumption for each protocol is increased with the data size of each source node. And for each case, the performance of our protocols is better than ExOR's.

To compare the packet loss rate, we set the data size of each source node equal to 500 and compared 18 source/destination pairs one by one for both protocols. The comparison results are shown in Fig. 7.

As we can see, the average packet loss rate of each pair increases as the hop count increases between a source and a destination node. For pairs with same hop numbers, the packet loss rate fluctuates due to the different unreliability of links and real-time traffic situation. In addition, in most cases, the packet loss rate is less than ExOR's.

The next comparison property is the end-to-end delay. We still let each source node send up to 500 packets towards its destination at the same time. We measure both average and max end-to-end delay time for each source/destination pair. Here, the definition of end-to-end delay of a packet is the time duration from a source node sent a packet to a destination received this packet. The average delay of each pair is illustrated in Fig.8 and the maximum delay for each pair is described in Fig. 9.

As we can see, the end-to-end delay of EEOR is smaller than EXOR's. This is mainly because in ExOR, a wireless node u sorts the neighbors nodes only by ETX (expected transmission count) when it chooses the forwarder list for a packet. However, the computation of ETX is not real time, when a node on some deliver path changed its ETX, other nodes may need to update their ETX one hop by one hop based on the new ETX value of this node.

In EEOR, for a wireless node u, we considered both the expected cost of a neighbor node v and the link error rate (which could be considered as real time) between u and v.

The last property we compared our protocol with ExOR is the packet duplication ratio. Here the main motivation to test the packet duplication ratio is that both our algorithm and ExOR are multi-path routing protocols. In most of cases, same packets will be relayed to the destination node through different pathes, thus increases the overhead of wireless networks. Even using other tricks like *Clique Method* or *Double ACK Method*, we still cannot guarantee that the packet will only arrive at the destination node at most once due to the unreliable links. Thus, multi-path property for unicast on the one hand decrease the packet loss ratio and energy consumption to some extend, on the other hand increase the overhead of the whole network. Fortunately, through our simulation results, the overhead increased by multi-path property is not much and the total energy consumption is decreasing as our conjecture. The reason is that for both protocols, a forwarder list for each engaging node constraints the area in which a packet can travel in the network, and eventually these multi-pathes will converge to some nodes or at least cross with each other. The result of duplication packet ratio is shown in Fig.6. Here, the definition of repeat times is the average times that a wireless node is required to forward how many duplicated packets for each source/destination pair.

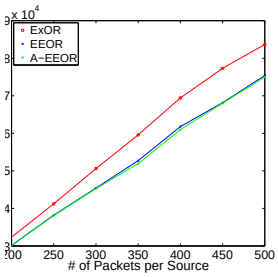


Fig. 3. Total transmissions.

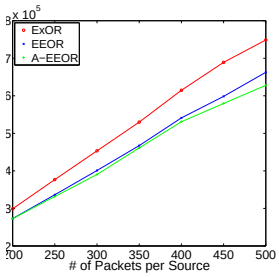


Fig. 4. Total received packets.

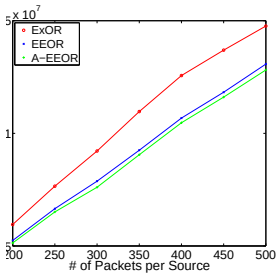


Fig. 5. Energy consumption.

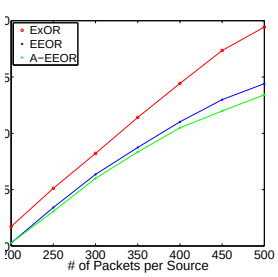


Fig. 6. Duplicated Packets.

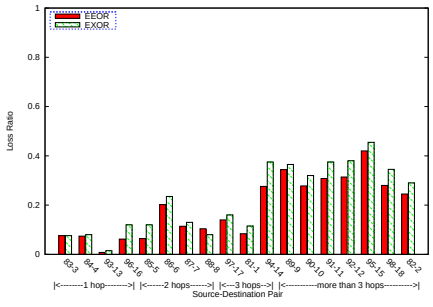


Fig. 7. Packet loss ratio.

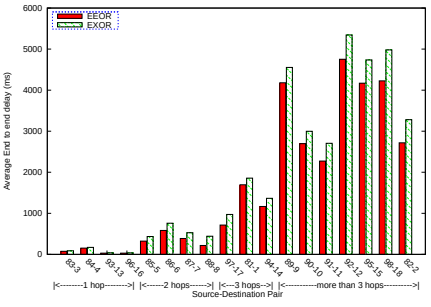


Fig. 8. Average delay for each pair.

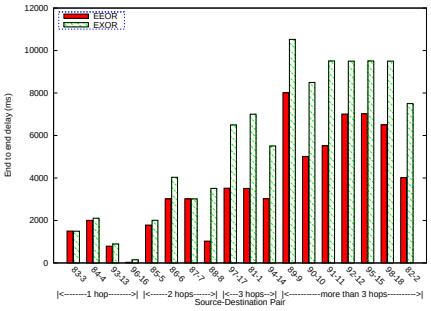


Fig. 9. Max delay for each pair.

6 Related Work

A number of energy efficient routing protocols [5], [12] have been proposed recently combining with a variety techniques. Most existing power aware protocols did not consider the packet losses of the wireless links. They assumed that the wireless links are reliable and then tried to theoretically provide performance guarantees [7], [13], [14].

There are some other protocols proposed recently to remedy the unreliability of the wireless channels such as using multi-path routing [9], [10], building reliable backbone [14], [8], and using energy efficient reliable routing structure [4], [19]. In [4], Dong and Banerjee addressed the problem of energy-efficient reliable wireless communication in the presence of unreliable or lossy wireless link layers in multi-hop wireless networks. Their main focus is on single path routing. Banerjee and Misra [19] explored the effect of lossy links on energy efficient routing and solved the problem of finding the minimum energy paths in the hop-by-hop retransmission model.

However, they all followed a conventional design principle in network layer of wired networks: after the best path(s) between a source and destination is calculated, all data flows from source and destination follow the selected path(s) until the path is updated after certain routing update period. ExOR [2] challenges this conventional design principle in network layer. MORE [3] presents a MAC-independent opportunistic routing protocol. MORE randomly mixes packets before forwarding them. This randomness ensures that routers that hear the same transmission do not forward the same packets. Thus, MORE needs no special scheduler to coordinate routers and can run directly on top of 802.11. Experimental results from a 20-node wireless testbed show that MORE's median unicast throughput is 22% higher than ExOR, and the gains rise to 45% over ExOR when there is a chance of spatial reuse. In addition to EXOR, [17] propose another opportunistic any-path forwarding protocol. Notice that ExOR and MORE were designed for large file transferring in wireless static mesh networks where energy saving is not a concern. Our protocol focused on minimizing the energy consumption of data forwarding in wireless sensor networks.

Recently [16] proposed a local metric, expected packet advancement (EPA) for GOR to achieve efficient packet forwarding. EPA for GOR is a generalization of EPA for traditional routing. Later, [15] proposed a new method of constructing transmission conflict graphs and proposed transmitter based conflict graph in contrast to link conflict graph.

7 Conclusion

Several interesting and challenging problems are left unsolved here. An interesting question is to design efficient protocols for selecting optimum forwarder list for multicast and broadcast. A challenge is to compute the expected cost accurately when we need to consider the additional overhead by sensor nodes for agreeing a unique node in the forwarder list to forward the data when multiple nodes could have potentially received the data correctly. It is interesting to design protocols using opportunistic routing that deliver the data most reliably, or deliver the data with the minimum delay.

REFERENCES

[1] X.-F. Mao, X.-Y. Li, W.-Z. Song, P. Xu and K. Moaveni-Nejad Energy Efficient Opportunistic Routing in Wireless Networks In *ACM MSWIM' 09*

[2] Sanjit Biswas and Robert Morris. Exor: opportunistic multi-hop routing for wireless networks. In *SIGCOMM*, pages 133–144, 2005.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *ACM SIGCOMM*, 2007.

[4] Qunfeng Dong, Suman Banerjee, Micah Adler, and Archan Misra. Minimum energy reliable paths using unreliable wireless links. In *ACM MobiHoc*, 449–459, 2005.

[5] Robin Kravets and P. Krishnan. Power management techniques for mobile communication. In *ACM MobiCom*, 1998.

[6] Johnson Kuruvila, Amiya Nayak, Ivan Stojmenovic. Hop count optimal position-based packet routing algorithms for ad hoc wireless networks with a realistic physical layer. *IEEE JSAC*, Vol. 23, No. 6, June 2005, 1267-1275.

[7] Xiang-Yang Li, Wen-Zhan Song, and Weizhao Wang. A unified energy-efficient topology for unicast and broadcast. In *MobiCom*, 1–15, 2005.

[8] Manki Min, Feng Wang, Ding-Zhu Du, and Panos M. Pardalos. A reliable virtual backbone scheme in mobile ad-hoc networks. In IEEE MASS, 2004.

[9] A. Nasipuri, R. Castaneda, and S. R. Das. Performance of multipath routing for on-demand protocols in ad hoc networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 6(4):339–349, 2001.

[10] J. Raju and J. Garcia-Luna-Aceves. A new approach to on-demand loop-free multipath routing. In *ICCCN*, pages 522–527, 1999.

[11] T.S. Rappaport. *Wireless Communications: Principles and Practices*. Prentice Hall, 1996.

[12] Volkan Rodoplu and Teresa H. Meng. Minimum energy mobile wireless networks. In *IEEE ICC*, volume 3, 1998.

[13] P.-J. Wan, G. Calinescu, X.-Y. Li, and O. Frieder. Minimum-energy broadcast routing in static ad hoc wireless networks. *ACM Wireless Networks*, 2002.

[14] Y. Wang, W.-Z. Wang, and X.-Y. Li, Distributed low-cost backbone formation for wireless ad hoc networks. In *ACM MobiHoc*, 2005.

[15] K. Zeng, W. Lou, and H, Zhai. On End-to-end Throughput of Opportunistic Routing in Multirate and Multihop Wireless Networks. In *IEEE InfoCom 2008*

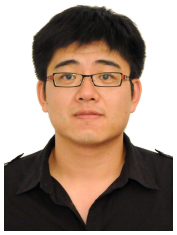
[16] K. Zeng, W. Lou, J. Yang, D. III. On geographic collaborative forwarding in wireless ad hoc and sensor networks. In *WASA 2007*

[17] Z. Zhong, J. Wang, and S. Nelakuditi. Opportunistic any-path forwarding in multi-hop wireless mesh networks. In *USC-CSE, Technical Report TR-2006-015*

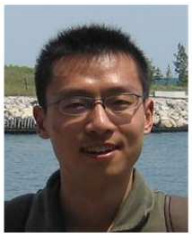
[18] H. Dubois-Ferriere, D. Estrin and M. Vetterli. Packet Combining in Sensor Networks In *ACM SenSys*, 2005.

[19] S. Banerjee and A. Misra. Minimum energy paths for reliable communication in multi-hop wireless networks. In *ACM MobiHoc 2002*.

[20] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *ACM MobiCom*, 2003.



Dr. Xufei Mao is a Computer Science PhD student at Illinois Institute of Technology. He received B.S. from Shenyang Univ. of Tech. and M.S. from Northeastern University, China. His research interests include design and analysis of algorithms for wireless networks and the design and implementation of large-scale wireless sensor networks. He is a student member of IEEE.



Shaojie Tang is a Computer Science PhD student at Illinois Institute of Technology. He received B.S. at Radio Engineering, Southeast University, P.R.China, 2006. His research field is on algorithm design, optimization, security of wireless networks, electronic commerce as well as online social network. He is a student member of IEEE.

Xiaohua Xu received the BS degree from ChuKochen Honors College at Zhejiang University, P.R. China, in 2007. He is currently working toward the PhD degree in Computer Science at Illinois Institute of Technology. His research interests and experience span a wide range of topics from theoretical analysis to practical design in wireless networks. He is a student member of the IEEE.

Dr. Xiang-Yang Li (M'99, SM'08) has been an Associate Professor since 2006 and Assistant Professor of Computer Science at the Illinois Institute of Technology from 2000 to 2006. He hold MS (2000) and PhD (2001) degree at Computer Science from UIUC. He received B.Eng. at Computer Science and Bachelor degree at Business Management from Tsinghua University, P.R. China in 1995. His research interests span wireless sensor networks, computational geometry, and algorithms, and has published over 200 papers and 4 books on these fields. He is an editor of IEEE TPDS, Networks: An International Journal, and was a guest editor of several journals, such as ACM MONET, IEEE JSAC. In 2008, he published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications". He is a senior member of the IEEE and a member of ACM.

Dr. Huadong Ma (M'99) received the B.S. degree in Mathematics from Henan Normal University in 1984, the M.S. degree in Computer Science from Shenyang Institute of Computing Technology, Chinese Academy of Science in 1990 and the Ph.D. degree in Computer Science from Institute of Computing Technology, Chinese Academy of Science in 1995. He is currently a Professor and Director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, Dean of School of Computer Science, Beijing University of Posts and Telecommunications, China. He visited UNU/IIST as research fellow in 1998 and 1999. From 1999 to 2000, he held a visiting position in the Department of EECS, The University of Michigan. He was a visiting Professor at The University of Texas at Arlington from July to September 2004, and a visiting Professor at HKUST from Dec. 2006 to Feb. 2007. His current research focuses on multimedia system and networking, Internet of things and sensor networks, and he has published over 100 papers and 4 books on these fields. He is member of IEEE and ACM.

APPENDIX

.1 Proof of Theorems

The proof of Theorem 1.

Proof: We prove this theorem by contradictions. Assume that the optimum forwarder list, say Opt, has the lowest expected cost and is not a prefix of $N^*(u) = \{v_1, v_2, \cdots, v_{|N^*(u)|}\}$. Then there must be two nodes $v_k, v_{k+1} \in N^*(u)$ such that $v_k \notin Opt$ and $v_{k+1} \in Opt$. Assume v_k is the first node satisfies conditions above when we check nodes in Opt^* (sorted list of Opt by increasing order) one by one. We will show that, if v_k is added to Opt, the expected cost decreases, i.e., $C_u(Opt^+) \leq C_u(Opt)$, and hence Opt cannot be the optimum forwarder list. Here $Opt^+ = Opt \bigcup \{v_k\}$.

First, let's examine a way to compute an upper bound, say Δ, of the expected cost for the node set Opt^+ as follows. If node v_{k+1} receives the packet and node v_k doesn't, then node v_{k+1} forwards the packet with cost $C_{v_{k+1}}$. If both v_{k+1} and v_k receive the packet, then node v_k forwards the packet with cost C_{v_k}. In other words, node v_k forwards data only if both node v_k and v_{k+1} receive the packet. It is obvious that this change increases the expected cost of node u, assuming same forwarder list Opt^+. In other words, $\Delta \geq C_u(Opt^+)$. Then, we compare Δ and $C_u(Opt)$. Let us consider the expected cost to forward the packet by one of the nodes in the forwarder list. The only time that the cost is different is when node v_k and node v_{k+1} both receive the packet and no node $v_i, i < k$ receives the packet. In the calculation of Δ, the cost is C_{v_k}; while in the calculation of $C_u(Opt)$, the cost is $C_{v_{k+1}}$ since $v_k \notin Opt$. According to the assumption, $C_{v_k} \leq C_{v_{k+1}}$, hence $\Delta \leq C_u(Opt)$.

Consequently, we have $C_u(Opt^+) \leq \Delta \leq C_u(Opt)$. That is to say, Opt can not be the optimum forwarder list, if it is not a prefix of $N^*(u)$. This completes the proof. □

The proof of Theorem 2.

Proof: Assume node u uses energy w to send the packet. For simplicity let $e = e_{uv_k}$, $C_1 = C_u(\text{Fwd}^*)$, and $C_2 = C_u(\text{Fwd}^* \bigcup \{v_k\})$. Using Equations (1), (3), and (5) we have

$$\begin{cases} (1-\alpha) \cdot C_1 = w + \beta \\ (1-\alpha \cdot e) \cdot C_2 = w + \beta + \alpha(1-e)C_{v_k} \end{cases} \tag{7}$$

Subtract and reorder, we have $(1-\alpha \cdot e) \cdot C_2 = (1-\alpha) \cdot C_1 + \alpha(1-e)C_{v_k}$. We know that $C_{v_k} < C_1$, so $(1-\alpha \cdot e) \cdot C_2 < (1-\alpha) \cdot C_1 + \alpha(1-e)C_1$, thus, $C_2 < C_1$.

Now we show that $C_{v_k} < C_2$. In Equation (7), we replace $(w+\beta)$ in the second equation with $((1-\alpha) \cdot C_1)$ from the first equation. Then we have $(1-\alpha \cdot e) \cdot C_2 = (1-\alpha) \cdot C_1 + \alpha(1-e)C_{v_k}$. We know that $C_{v_k} < C_1$, thus, $(1-\alpha \cdot e) \cdot C_2 > (1-\alpha) \cdot C_{v_k} + \alpha(1-e)C_{v_k}$. This implies that $C_2 > C_{v_k}$. □

The proof of Theorem 3.

Proof: The proof is similar to that of Theorem 2. In Equation (7), we subtract and reorder to get $(1-\alpha \cdot e) \cdot C_2 = (1-\alpha) \cdot C_1 + \alpha(1-e)C_{v_k}$. We know that $C_{v_k} > C_1$, thus, $(1-\alpha \cdot e) \cdot C_2 > (1-\alpha) \cdot C_1 + \alpha(1-e)C_1$. This implies that $C_2 > C_1$. □

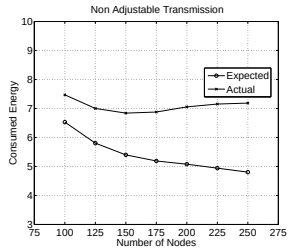
.2 Concept Validation by Simulations

Besides we design the system and compare its performance to other protocols in TOSSIM, we also run the following simulations in self-designed simulator (which does not consider radio interferences). We randomly placed 100, 125, 150, 175, 200, 225, and 250 nodes in a $50feet \times 50feet$ square region, the transmission range of each sensor node is set to $10feet$. For each set of nodes we generated 100 random graph, for each random graph we tried 100 random pair of source and target nodes, and for each pair of source and target nodes we tried the routing 100 times. In addition to that, for each set up, we tried the cases where the average error producibility of links were 0.25, 0.50, 0.75, and the extreme case 0.95 and then we run each set up on 6 different methods for both non-adjustable and adjustable transmission models.

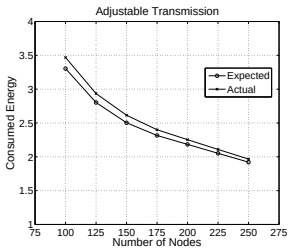
The routing and forwarder selection method we compared includes:

- **Standard Dijkstra** we implemented two variation of Dijkstra. In non-adjustable transmission model we assign the weight $\frac{1}{1-e}$ (e denotes the error probability of a link) to each node and then run the traditional Dijkstra and in adjustable transmission model we assign the weight $\frac{w}{1-e}$ to each link and then we run the traditional Dijkstra.
- **Distance Based:** each node selects a certain number of its neighbors that are closest to the destination node. Actually, this naive method can be improved by the idea in [6] which selects a certain number of neighbors based on cost function $w/(1-p)$, this is referred as *Cost function Based method* ([6]). Notice, the results of simulation refer to Distance Based method rather than the method in [6].
- **Reception Based:** each node selects a certain number of its neighbors that have the best reception (*i.e.*, small error probability).
- **Most Forwarding:** this method is also known as *compass forwarding*. Each node selects a certain number of its neighbors that are the most forwarding nodes. Briefly if a message is located at a node v, and wants to reach node t, most forwarding will send it to the neighbor u of v such that the slope of the line segment joining u to v is the closest to the slope of the segment joining v to t.

Previously, we calculated the expected costs of the nodes based on the assumption that only one among the nodes in the forwarder list forwards the packet. But our routing method does not guarantee that. In our routing method each node sends ACK messages to its neighboring nodes to tell them not to forward, but what if there is another node in the forwarder list that is not in the transmission range of the sender of the ACK message? For example, a node u sends a packet to nodes v and w which are both in the forwarder list of node u. If the $\angle vuw > 60$ then nodes v and w will not receive ACK messages from each other under non-adjustable model. So if nodes v and w both receive the packet, they both will forward. This is the reason why the actual cost of our routing method

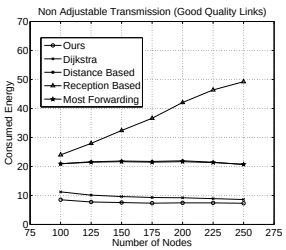


(a) non-adjustable power

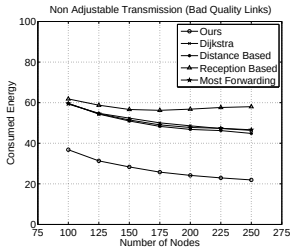


(b) adjustable power

Fig. 10. Expected cost vs actual cost.



(a) good links



(b) bad links

Fig. 11. Energy consumption comparison.

is slightly higher than the expected cost. In the adjustable transmission model, nodes tend to choose nearby neighbors as forward nodes since they usually reduce their power for power saving purposes, so the above case does not happen very often. On the other hand, in the non-adjustable transmission model, nodes tend to select bigger forward lists and hence the above case happens more. The Figure 10 (a) and Figure 10 (b) show the difference between the actual cost and expected cost for adjustable and non-adjustable transmission model separately.

.2.1 Non-Adjustable Transmission Model

As can be seen in Figure 11 our method outperforms distance based, reception based, and most forwarding methods. If links are reliable, then Dijkstra will perform very close to the optimum, so there is not much room for improvement, but if links have very poor reliability, specially when the graph is dense, then our method will also outperform Dijkstra. See Figure 11 for an illustration.

.2.2 Adjustable Transmission Model

The same arguments apply to the adjustable transmission model. As can be seen in Figure 12 our method again outperforms distance based, reception based, and most forwarding methods. If links are reliable, then Dijkstra will perform very close to the optimum, so there is not much room for improvement, but if links have very poor reliability, especially when the graph is dense, then our method will also outperform Dijkstra. See Figure 13 for an illustration.

.2.3 Source-Target Distance

We can look at the problem from another point of view. We can compare the energy consumed to route a packet based on the distance between source and target. As can be seen in figure 14 all methods perform closely when the target is close (*i.e.*, less than one hop away), but the difference increases as the distance increases.

Figure 15 illustrate different forwarder list selection methods implemented and discussed in this paper. We randomly placed 100 nodes in a $50feet \times 50feet$ square region and the transmission range of each sensor node is set to $10feet$.

.2.4 Further Improvement

As mentioned before the expected cost calculated at each node is based on the assumption that only one node in the forwarder list forwards the message. This is the ideal case and the routing algorithm introduced in this paper cannot guarantee that. The reason is that the nodes in the forwarder list do not necessarily build a clique. If two nodes in the forwarder list cannot hear each other, both of them will forward the message. To alleviate this problem we introduce two additional methods:

Double ACK method: in this method when the sender receives the ACK message from one of the nodes in its forwarder list, it sends a second ACK message to its neighbors whose expected cost is higher than the expected cost of the node that sent the first ACK message. This will guarantee that only one node in the forwarder list forwards the message if the ACK is free of lost.

Clique method: in this method we remove some nodes from the forwarder list such that the forwarder list of each node is a clique. Consider node u and its forwarder list, and assume node v has the smallest expected cost among the nodes in $\mathrm{Fwd}(u)$. Then for each node w in $\mathrm{Fwd}(u)$, we keep node w iff the following two conditions are satisfied. 1) $\angle vuw < 30$; 2) nodes v and w are neighbors of each other. Otherwise we delete node w from $\mathrm{Fwd}(u)$. Unlike all other algorithms and method introduced in this paper, Clique method requires some geometry information to be able to calculate the angle. This guarantees that all the nodes in the forwarder list are contained in a pie whose angle is at most 60 and hence the nodes in the forward list will make a clique. Note that removing nodes from forwarder list will increase the expected cost at the node, but on the other hand, guarantees that only one node forwards the message.

To emphasize on WBA property and the impact of the $DoubleAck$ method and $Clique$ method placed the same number of nodes in a $30feet \times 80feet$ rectangular region and we also placed the source node and the target node at the opposite ends of the region.

In Figure 16 the average error rate is 30%, so there is a good chance that two or more nodes in the forwarder list forward the message. So our method consumes more energy than the Dijkstra method. The Clique method and the Double ACK method perform very close to optimum.

In Figure 17 the average error rate is 70%, so there is less chance that two or more nodes in the forwarder list forward the message. So our method consumes less energy than the Dijkstra method. Again the Clique method and the Double ACK method perform very close to optimum.

In Figure 18 the average error rate is 95%, so there is very little chance that two or more nodes in the forwarder list forward the message. So our method not only outperforms

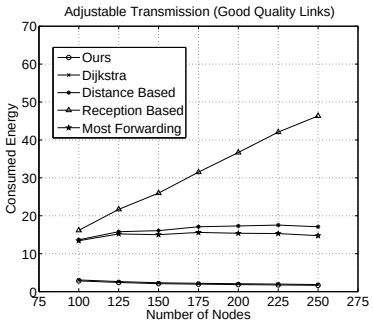


Fig. 12. Energy consumption comparison. Good links

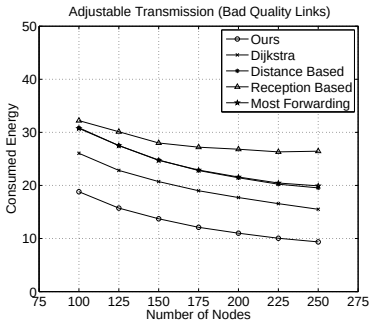


Fig. 13. Energy consumption comparison. Bad links

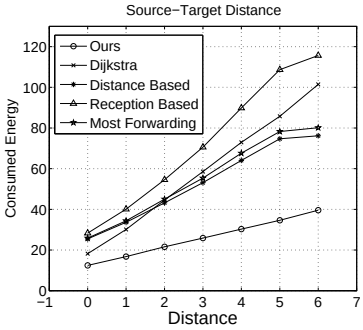
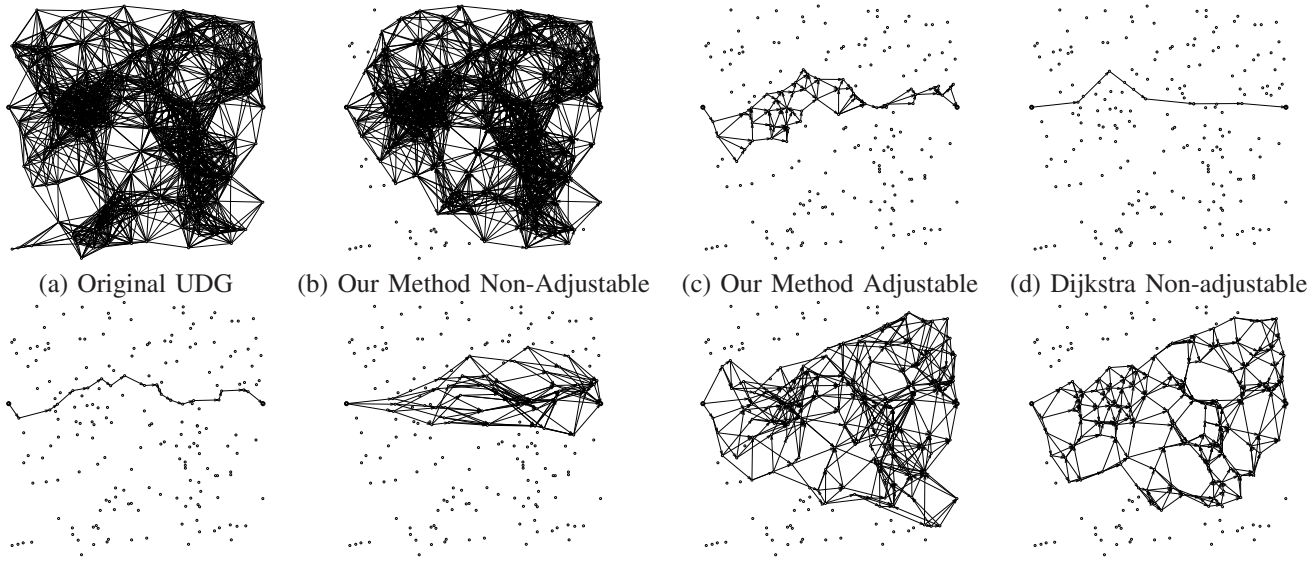


Fig. 14. Energy consumption comparison based on source-target distance



(a) Original UDG (b) Our Method Non-Adjustable (c) Our Method Adjustable (d) Dijkstra Non-adjustable

(e) Dijkstra Adjustable (f) Distance Based (g) Reception Based (h) Most Forwarding

Fig. 15. Different forwarder list selection methods

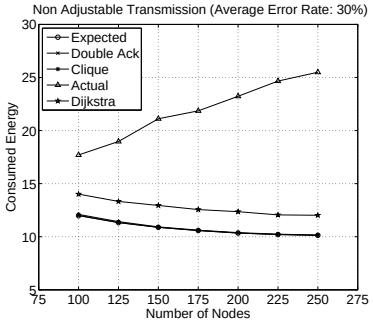


Fig. 16. Average Error Rate: 30%

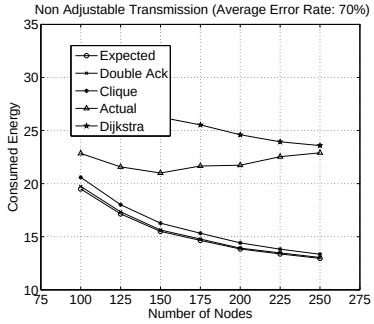


Fig. 17. Average Error Rate: 70%

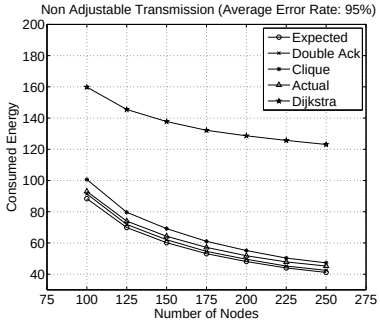


Fig. 18. Average Error Rate: 95%

Dijkstra method but also it performs slightly better than the Clique method because the Clique method removes some of the nodes from the forward list.

.3 System Design and Implementation

Table 1 introduces some main properties that a wireless node and a packet are related with.

TABLE 1
Properties related with wireless node u and packet p

$u.id$	the unique identity of u
$p.src$	source node identity of p
$p.des$	destination node identity of p
$ACK(p)$	ACK message corresponding to p
$u.fwdList(p)$	forwarder list that u constructs for p
$u(p).pri$	priority of u to forward p, this is indicated in the forwarder list contained in p
$LER(u,v)$	link error rate from u to its neighbor node v
$LR_i^\tau(u,v)$	i type packet loss rate from u to v during the past τ time, here $i \in \{data, beacon, ack\}$ denotes three different message type
$N_i^\tau(u)$	# of i type packets sent by u during the past τ time, $i \in \{data, beacon, ack\}$
$u.pktList$	Packet list of u, which records information of all data packets u received during a period of time.

.3.1 Node data structures

We assume every node has a unique node identity. In our design, a wireless node u could act as several roles simultaneously, like a source node, a destination node and a relay node for one or more different source/destination pairs. The function for each role is very simple. For example, when u acts as a source node, u will construct and send new packets periodically. When receiving a new packet p, a node u will compute $u.fwdList(p)$ first and then broadcast p to all neighbors when u is not the destination node of packet p. When u's role is a relay node for packet p, it will replace the old forwarder list contained in p with new forwarder list computed by u and broadcast p. If u works as a destination node, it simply get data payload of p (may do further operation depending on different system).

After a node u booted, u only needs to know (collect) the information of its one hop neighboring nodes. This is conducted by listening to the beacon, data and ACK messages from neighbors. In addition, u will maintain a routing table which contains necessary information for u to compute the forwarder list for a packet p to p's destination. Simply speaking, if a node u wants to send or relay a packet p to its destination v, what u needs to do is to compute the optimal forwarder list $p.fwdList$ based on the current information in the routing table, and broadcast p to all nodes in $p.fwdList$. To sum up, the detail operations of u to do this mainly depend on following aspects.

- *Routing Table*: which records its expected cost to forward a packet to the destination and the expected cost of each neighbor to relay this packet to each specific destination. These information will be used to compute a node's real-time forwarder list to the specific destination. The routing table will be updated continuously by collecting information from all data, beacon and ACK messages from neighbors.
- *Buffer Size*: which determines directly how many packets a node can handle simultaneously, thus plays a key role for a node to make further decision when it generates or is required to relay a packet.
- *Priority* : which indicates on which priority a node should forward (send) a packet. This priority is assigned by the sender and indicated by the order, following which a receiver is in the forwarder list. The source node has the highest priority for all packets it generated automatically, *i.e.*, a node will take care its own packets first when conflict between its own packets and relay packets happens.
- *Packet List*: which records the unique information (e.g. sender ID + packet ID) of all packets a node listened; Before a node takes care of a packet, it will look up the Packet List first and check out what operations the node have done to previous seen packets during a period, then decide the next step.

More detailed definition and explanation of aforementioned items are shown in C.4. For simplicity, from now on, we focus on one destination node.

.3.2 Link Reliability Measurement

Another issue needs to be considered carefully is the link reliability of EEOR since the link reliability is directly related with the choice of a node's forwarder list, thus determines the performance of EEOR. As we know, the link reliability between two neighboring nodes not only depends on the transmission power of the transmitter, background noise, but also depends on its logical position in the network. For example, if a node is surrounded by many source nodes, the link reliability from it to its neighbors probably could be worse than other nodes due to the numerous competition for radio since data source nodes will push new data to the network continuously.

In our design, besides considering link gain, power, and background noise, we considered several other aspects to compute the link reliability between two neighboring nodes as well. We considered the transmission situation of all three types of messages together during a period of τ time ($\tau = 1000\ ms$ in our design) to get the link reliability for two wireless neighboring nodes since both their packet length and utilized frequency are different. For two neighbors u and v, the way to compute the current link error rate $LER(u,v)$ is as follows: After node u started, u will continue to collect the information from data, beacon and ACK messages received from v for the past τ time and update the current packet loss ratio for data ($LR_{data}^\tau(u,v)$), beacon ($LR_{beacon}^\tau(u,v)$) and ACK ($LR_{ack}^\tau(u,v)$) messages respectively. Notice, every node will also broadcast how many packets it has broadcast and how many packet it has received successfully from its neighboring nodes during current time window. For example, u collects the neighboring information (e.g. ID, expected cost for forwarding a packet to the destination, data transmission successful ratio, etc.) from beacon message, collects packets

from data message by receiving or overhearing, collect sending status from ACK messages and so on. Thus, the current link error rate satisfies:

$$LER(u,v) = \sum_{i\in\{data,beacon,ack\}} \frac{LR_i^\tau(u,v) \times N_i^\tau(u)}{\sum_{j\in\{data,beacon,ack\}} N_j^\tau(u)}$$

.3.3 Forwarder List

When a node u prepares to send a packet p to its destination node $p.des$ (no matter p is generated by u or needs to be forwarded by u), u have to compute the current forward list of p first by Algorithm 1 or 2 for two different models respectively. The forwarder list is updated in real-time and could be different according to the different destinations for a given time. Given a node u and a packet p, the construction of $u.fwdList(p)$ is determined by considering several factors:

- Link error rates between u and nodes in $N(u)$;
- Expected cost of each neighbor to forward p to $p.des$;
- Traffic situation of each neighbor;

Here, a node's traffic situation will be considered when a node computes its expected cost to a specific destination, this will be explained later in *Congestion Handling* subsection (supplementary file). One thing needs to be mentioned that there is a tradeoff between the frequency to update the forwarder list and the frequency of a node to broadcast beacons to its neighbors. When a node collects "enough" information from messages it received or after a specific time period, it will re-compute its forwarder list for each potential destinations and update its expected cost for every destinations. Otherwise, the forwarder list updating either wastes resources or does not reflect the optimal forwarder list real time.

.3.4 Message Handling

As we have mentioned before, after a node receive a packet, it may save it, drop it, forward it immediately or waits for a while to make further decision. In this subsubsection, we will explain the details of the operation when a node receive beacon, ACK and data message respectively.

Every node will broadcast beacon messages periodically which will be operated by the routing layer. After receiving a beacon message from one of neighboring nodes, a wireless node will collect necessary information from the beacon and update its known information.

Before presenting our strategies to handle ACK and $data$ packet, we need to introduce a concept named *Packet List* because it is important and will be used in both strategies. *Packet List* records some necessary information of the packets a node received during a period, like source identity, sequence number, and so on. Notice, *Packet List* only records information of a packet, rather than packet itself due to the limited storage of a wireless node.

After a node u receives a packet p, u will broadcast an ACK message after waiting for t times if u belongs to the forwarder list that is contained in p. Here t depends on node u's priority (remembering that this priority is determined by the order in the forwarder list of p), the higher the priority of node u is, the shorter t is. The basic idea for a node to react upon receiving an ACK message is as follows: when the sender node of p receives ACK, it will stop sending p again. If another node v which belongs to the forwarder list of p receives ACK(p), v will compare its priority to forward p with u's. v will drop p if v finds it has lower priority. The node in the forwarder list which has highest priority will become the new source node of p and continue to forward p to p's destination. To guarantee the high data transmission success ratio, a sender will retransmit a packet for several times (within the upper-bound which is 5 in our simulation) until it receives ACK or overhears the packet being forwarded by some node in the forwarder list. We set up the same threshold for all algorithms compared in the simulation section in order to guarantee the fairness. Please see Algorithm 5 for more details. In Alg. 5, we use abbreviations $\{$ *Pd*, *Pf*, *Po*$\}$ to denotes the status of a packet (saying p) for a node (saying v). Here, $p.status$ == *Pd* means p has been delivered successfully by v; *Pf* means p will be forwarded by v later. *Po* means p is created by v originally.

Thus all or most of nodes in the same forwarder list including the sender node itself will agree on which node(s) will act as new source node(s) of this packet upon receiving ACK on time. However, nodes in the same forwarder list may not be able to receive message from each other. For example, they are not neighbors or the ACK message could be lost. There will be several nodes that become new senders of p sooner or later and increase the duplication of p and traffic burden. Actually, based on our observation through running the case on the TOSSIM, both cases aforementioned happened frequently with the increment of the traffic and the increment of unreliability between links. In order to mitigate this situation, we design the "snoop" mechanism. Basic idea is to let a wireless node to snoop the message from its neighbors and analyze the situation by comparing the information of new coming message with the information recorded in its *Pack List* before making further decision. Details please refer to Algorithm 5. Our simulation results indicate that the *Message Handling* algorithm can decrease the duplication of packets significantly and reduce the packet delay as well.

Our simulation results showed this packet handling mechanism can mitigate the duplications of packets extremely in most of cases. We also take the congestion handling problem into consideration.

.3.5 Congestion Handling

When the traffic increases to the network capacity, some packets will be dropped by some nodes due to the limited buffer size. For example, the source nodes generates packets so quickly that inter-relay nodes cannot forward all packets on time and do not have enough space to save more packets into buffer at the same time, many packets will be dropped. In addition, the decreasing of the link reliability because of the weather or some other reasons will lead to the increase of retransmission times, thus degrade the performance. Thus, a source node should have the ability to adjust its speed to initiate sending new packets to the network. We use a similar strategy as TCP to handle the congestion related issue. Recall each node maintains a real-time routing table towards each specific destination. If a node u finds some of its links'

Algorithm 5: Message Handling

```
input : Receiving a packet p
begin
    if p is a data packet then
        if this is the first time for u to receive p then
            if u.id == des(p) then
                broadcast(ACK(p)); save data packet p;
                pktList(p).status = Pd;
            if u.id == sdr(p) then
                Drop(p); pktList(p).status = Pd;
            if u.id ∈ p.fwdList then
                broadcast(ACK(p));
                putPacketInBuffer(p); and
                pktList(p).status = Pf;
            else
                pktList(p).status = Pd;
        else
            if pktList(p).status == Pd then
                Drop(p);
            if pktList(p).status == Po then
                Drop(p); pktList(p).status = Pd;
            if pktList(p).status == Pf then
                if sdr(p).pri ≥ u(p).pri then
                    Drop(p); pktList(p).status = Pd;
                else
                    Drop(p);
            else
                Drop(p);
    else
        if p is ACK(q) then
            if q ∉ u.pckList or
            u.pckList(q).status == Pd then
                Drop(p);
            else
                if u.id == q.sdr or
                (p.sdr)(q).pri > u(q).pri then
                    u.pckList(q).status=Pd; Drop(q);
        else
            Get information from beacon message p, and
            update local information;
end
```

reliability is decreasing (resp. increasing) or congested, the computed expected cost will increase (resp. decrease). The increasing (or decreasing) of the expected cost of u will lead to the increment of expected cost of a series of nodes (probably all nodes from u to the source node on one of multiple pathes), finally this will either make the packets toward specific destination choose different routing pathes (or push more packets to this path) or lead the source node to increase (or decrease) its expected cost. When a source node finds its expected cost is increasing (resp. decreasing) (more than 25% in our simulation) it will decrease (resp. increasing) the data rate by half. Our simulation results indicate that our strategy can handle the congestion well.

.4 Description of ExOR [2]

ExOR broadcasts each packet, choosing a receiver to forward only after learning the set of nodes which actually received the packet. Delaying forwarding decisions until after reception allows ExOR to try multiple long but radio lossy links concurrently, resulting in high expected progress per transmission. The key challenge in realizing ExOR is ensuring that only the best receiver of each packet forwards it, in order to avoid duplication. ExOR operates on batches of packets in order to reduce the communication cost of agreement. The source node includes in each packet a list of candidate forwarders prioritized by closeness to the destination. Receiving nodes buffer successfully received packets and await the end of the batch. The highest priority forwarder then broadcasts the packets in its buffer, including its copy of the "batch map" in each packet. The batch map contains the sender's best guess of the highest priority node to have received each packet. The remaining forwarders then transmit in order, but only send packets which were not acknowledged in the batch maps of higher priority nodes. The forwarders continue to cycle through the priority list until the destination has 90% of the packets. The remaining packets are transferred with traditional routing.