Maximum Residual Energy Routing with Reverse Energy Cost

Qiling Xie†, Chin-Tau Lea†, Mordecai J. Golin‡ and Rudolf Fleischer‡
†Dept. of Electrical & Electronic Engineering
Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong
{sieh, eelea}@ee.ust.hk
‡Dept. of Computer Science
Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong
{golin, rudolf}@cs.ust.hk

Abstract-**The Maximum Residual Energy Path (MREP) routing has been shown an effective routing scheme for energy conservation in a battery wireless network. Past studies on MREP are based on the assumption that the transmitting node consumes power, but the receiving node does not. This assumption is false if acknowledgement is required, or if the ad hoc network has deployed the energy-conservation mode (sleeping mode). When backward energy consumption is present in transmission (i.e. the receiving end consumes energy), finding an MRE path that has enough energy for finishing the transmission has become NP-hard. We show in this paper a Dijkstra-like heuristic algorithm for finding the optimal MRE path. The new algorithm guarantees that once a path is found, it will have enough energy to finish the transmission task, while the original MREP algorithm, ignoring the backward energy costs, cannot guarantee that. We also show another routing technique that can extend the system life. The technique works for both MREP-based routing schemes.**

I. INTRODUCTION

Recent advances in wireless technologies, such as Bluetooth [1], have made it easy and practical to construct an ad hoc network [11] for novel applications – a surveillance network, a wireless tag network in a grocery store, and a sensor network [2] to monitor environment dangerous conditions are just a few examples.

In such a network, battery power is a vital resource for each wireless device (except for the wired gateway computers), and energy conservation is a critical issue. Routing plays an important role in energy conservation. This issue has been studied extensively in the past. A central part of any routing study is the definition of the path metric. Some metrics combines both delay and power consumption, whereas others focus on maximizing the system life and ignore the delay.

When delay is less a concern than system life, the Maximum Residual Energy Path (MREP) routing has been shown an effective scheme for energy conservation [3,4]. In MREP routing, the best path is one that has the maximum residual energy left after sending the message. Past MREP routing is based on the assumption that only the transmitting node consumes energy, but the receiving node does not. This assumption is not true if acknowledgements are required, or if the receiving node needs to be waken up from its energy conservation mode (sleeping mode). In either case, the process of receiving a packet requires the receiving node send some thing back to the sending node and that will consume energy at the receiving side.

When backward energy consumption is present in transmission (i.e. the receiving end consumes energy), finding an MRE path that has enough energy for finishing the transmission of a packet has become NP-hard [10]. We show in this paper a Dijkstra-like heuristic algorithm for finding the optimal MRE path. Compared with the old MREP algorithm that ignores the backward energy consumption, the new algorithm can guarantee that once a path is found, it will have enough energy to finish the transmission task. The original MREP algorithm, however, cannot guarantee that. In addition, we show another technique that can extend the system life for MREP-based routing scheme. This method works for both the original MREP and the new modified MREP.

The rest of the paper is organized as follows. Section II gives the definition of the problem. Section III shows the original MREP algorithm and the new heuristic MREP algorithm is introduced. Simulations and results are presented in Section IV. Finally, some concluding remarks are made in Section V.

II. DEFINITIONS

The routing problem can be modeled as a directed graph $G = (V, L)$, where V is a set of vertices (wireless nodes) and L is a set of directed connections (u, v) from a vertex u to another vertex v. Initially, each vertex u has a battery charged with energy $E_u \geq 0$. This energy will be consumed by transmitting messages. $E(u)$ is the current battery energy available at node u before routing a message.

With each edge $e=(u, v)$, we associated two costs, the sending cost s_e(or $s_{u,v}$) for sending a message along e, and the acknowledgement cost r_e (or $r_{u,v}$) for receiving a message. Sending a message along e will reduce $E(u)$, the energy of u, by s_e, and it will reduce $E(v)$ by r_e. Of course, sending a message is only possible if none of the residual energies becomes negative.

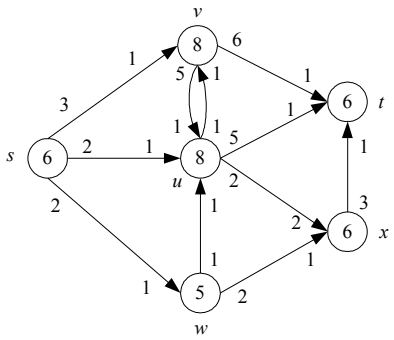

Fig. 1. A routing network. Nodes are labeled by their battery energy. Outgoing edges are labeled by the sending cost along the edge, and incoming edges are labeled by the acknowledgement cost.

Normally, s_e and r_e will depend on the distance between the vertices and on the size of the messages, but that is not the focus of our discussion.

When sending a message from a vertex s to another vertex t in G, the message can be routed along one of various paths. Of interest are paths that leave a high energy level in all the nodes, i.e. situations in which routing a message would use up all the energy of one single node should be avoided. If this is not done, trouble might arise when the next message is routed. Further, the network could even become disconnected.

Formally, if a message is routed along a path $P=(v_1, v_2,\ldots, v_{k-1}, v_k)$ in G, where $v_1,\ldots, v_k$ are vertices and $(v_1, v_2),\ldots, (v_{k-1}, v_k)$ are the edges, then each node on P loses a certain amount of energy due to the sending cost and the acknowledgement cost. Let R_u denote the *residual energy* of u after routing a message. If P is simple then

$$R_{v_i} = \begin{cases} E(v_1) - s_{v_1,v_2} & i = 1 \\ E(v_i) - r_{v_{i-1},v_i} - s_{v_i,v_{i+1}} & i = 2,\cdots,k-1 \\ E(v_k) - r_{v_{k-1},v_k} & i = k \end{cases} \quad (1)$$

If P is not simple, then it may contain some vertex several times; in this case, all the sending and acknowledgement costs for that vertex have to be added up. We now define the Minimum Residual Energy (MRE) of P, denoted by $D(P)$, to be

$$D(P) = \min_{i=1,\ldots,k} \{R_{v_i}\} \quad (2)$$

Note that all residual energy values must be non-negative. If they are not, the message cannot be sent. A path P with $D(P)\geq 0$ is called a *legal path*. For example, in Fig. 1, consider the path $P=(s, w, u, x, t)$. The residual energy left in s is $R_s = E(s) - s_{s,w} = 6 - 2 = 4$. Similarly, it can be seen that $R_w = 3$, $R_u = 5$, $R_x = 1$, and $R_t = 5$. The path therefore has MRE $D(P)=1$. The non-simple path $P'=(s, u, v, u, t)$ leaves $R_s = 6 - 2 = 4$ in s, $R_u = 8 - 1 - 1 - 1 - 5 = 0$ in u, $R_v = 8 - 1 - 5 = 2$ in v, and $R_t = 6 - 1 = 5$ in t. Thus, $D(P')=0$.

A path is *optimal* if it has a maximum MRE among all paths that route a message from a given start vertex to a given end vertex. Routing along a non-simple path is obviously never a good idea, so optimal paths are always simple. Obviously, the problem that needs to be solved is listed in Fig. 2.

Maximum Residual Energy Path (MREP) Problem

Input:
A directed graph $G = (V, L)$ with vertex energies E_u, sending costs $s_{u,v}$, and acknowledgement costs $r_{u,v}$. Also, two vertices s (source) and t (destination).

Output:
An $s - t$ path in G that maximizes $D(P)$ among all $s - t$ paths P in G.

Fig. 2. The Maximum Residual Energy Path problem

III. Modified MREP Algorithm

A. MREP without Backward Energy Consideration

Let $G = (V, L)$ be a directed graph such that associated with each edge $(u,v)\in L$. There is a *bandwidth* $b(u, v) \geq 0$. Now let $P=v_1, v_2,\ldots, v_{k-1}, v_k$ be a path in G. The *bottleneck cost* $B(P)$ of the path is defined to be

$$B(P) = \min_{i=1,\ldots,k} b(v_i, v_{i+1}) \quad (3)$$

Intuitively, $B(P)$ is the maximum amount of available bandwidth to send the message from v_1 to v_k along path P.

Given $s,t\in V$, the *max-bottleneck problem*, sometimes known as the *max-bandwidth problem*, is to find a path from s to t that maximizes $B(P)$. This problem has been extensively studied and it is well known that such a path can be found using a simple modification of Dijkstra's shortest path algorithm [9], which builds a tree rooted at s in which the unique path from s to t is a max-bottleneck path between s and t. We listed the modified Dijkstra-like algorithm in Fig. 3. It builds a tree T rooted at s in which s the parent of node w is stored in $P[w]$. The optimal path connecting s and t can be read off backwards by starting at t and looking at t, $P[t]$, $P[P[t]]$, etc. until s is reached. The standard implementation of this algorithm runs in

Modified Dijkstra's algorithm for finding max-bottleneck paths

Input: Source s and destination t.
Output: A max-bottleneck spanning tree rooted at s containing a path to t.

1. **for** every node $v\in G$ **do**
   $\{P[v] = 0; C[v] = -\infty\}$
2. $C[s] = +\infty$; $F = \phi$.
3. **for** every neighbor w of s **do**
   $\{P[w] = s; C[w] = b(s, w)$; add w to $F\}$
4 **repeat**
   remove the node with maximum $C[u]$ from F;
   for every neighbor w of u **do**
   if $C[w]= -\infty$
   $\{P[w] = u; C[w] = \min(C[u], b(u, w))$; add w to $F\}$
   else if $\{w\in F$ and $C[w]<\min(C[u], b(u, w))\}$
   $\{P[w] = u; C[w] = \min(C[u], b(u, w))\}$
until $C[t] \neq -\infty$ and $t \notin F$.

Fig. 3. The Modified Dijkstra's algorithm (based on [10]). $C[w]$ will be the cost of an optimal (max) bottleneck path between s and w. The algorithm builds a tree (stored using the predecessor $P[w]$ links) such that the unique path in the tree from s to w is an optimal $s - w$ path.

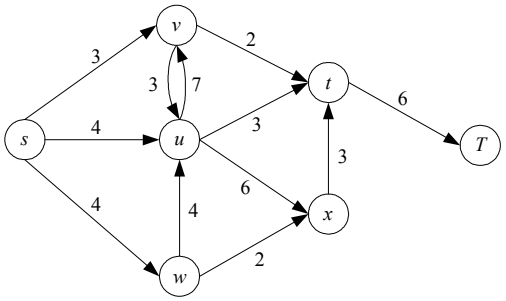

Fig. 4. An example graph with bandwidth costs on edges. There are many max-bottleneck $s - T$ paths all having cost 3. One such path is s, u, t, T. Note that in this path the bottleneck is edge (u, t).

$O(|L|\log|V|)$ time; with more sophisticated data structure this can be reduced to $O(|L|+|V|\log|V|)$ time [10,11].

We now discuss how to find optimal Maximum Residual Energy paths from s to t when $r_{i,j} = 0$, i.e. there is no acknowledgement costs for messages. The algorithm is actually a very simple transformation of the problem that takes the energy costs from the vertices and puts them on the edges and then runs the max-bottleneck algorithm. The reason that we describe it in detail is as an introduction to the more complicated transformation needed in the following parts when $r_{i,j} \neq 0$.

Rewriting the problem for the $r_{i,j} = 0$ case, we want to find $s - t$ paths P: $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$ that maximizes

$$D(P) = \min\left(E(v_k), \min_{1 \le i < k}\left(E(v_i) - s_{v_i, v_{i+1}}\right)\right) \quad (4)$$

We can now see how to transform the MREP problem into a bottleneck one. The idea is to introduce a new graph $G'=(V', L')$ which is exactly G with one new vertex T and one new link (t, T) added. The $b()$ for G' is defined by setting $b(t, T)=E(T)$ and for all other edges $(u, v) \in L'$ setting

$$b(u, v) = E(u) - s_{u,v} \quad (5)$$

As an example, if G is the graph in Fig.1 but without acknowledgement cost on each incoming edge, then the transformed new graph G' is depicted in Fig. 4. We first note that there is a one-to-one correlation between $s - t$ paths in G and $s - T$ paths in G'. If P: $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$, the associated path in G' will be P': $s = v_1, v_2, \ldots, v_{k-1}, v_k = t, T$. Since (t, T) is the *only* link entering T, this correlation is reversible, i.e., if P': $s = v_1, v_2, \ldots, v_{k-1}, v_k = t, T$ is an $s - T$ path in G' then this is associated with the $s - t$ path P: $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$ in G. For example, the path P: s, u, t in Fig. 1 corresponds to P': s, u, t, T in Fig. 1.

The reason why this correlation is important is that if P: $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$ is a simple path in G then

$$D(P) = \min\left(E(v_k), \min_{1 \le i < k}\left(E(v_i) - s_{v_i, v_{i+1}}\right)\right) = B(P') \quad (6)$$

Example: Refer again to Fig. 1 and Fig. 4, P: s, u, t has $D(P)=3$ and the associated P': s, u, t, T also has $B(P')=3$.

Recall that algorithm in Fig. 3 finds the max-bottleneck path in G'. Furthermore, the paths that it finds are simple since once a node u appears on a path, it is removed from F and never appears again. Thus, the algorithm in Fig. 3 finds the maximum bottleneck $s - T$ simple path in G' which corresponds to the

1. Transform $G = (V, E)$ into $G' = (V', E')$ by adding vertex T and edge (t, T).
2. Use the modified Dijkstra's algorithm from Fig. 2 to find P', a max-bottleneck $s - T$ path in G'.
3. If P' is not a simple path, chop off cycles to make it a simple $s - T$ path.
4. If P': $s = v_1, v_2, \ldots, v_{k-1}, v_k=t$, T let P be $s = v_1, v_2, \ldots, v_{k-1}, v_k=t$
5. Return P.

Fig. 5. Algorithm for finding MRE $s - t$ path when $r_{i,j}=0$ for any i, j.

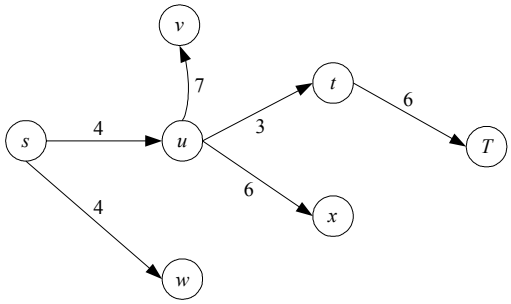

Fig. 6. A spanning tree output by the modified Dijkstra's algorithm from the graph in fig. 3. Note that every path in the tree from s to a node is a max-bottleneck path to that node.

MRE $s - t$ simple path in G. If this is found, then an algorithm for finding MRE paths given in Fig. 5 can be determined.

As an example, starting with G as given in Fig. 1 (again, no acknowledgement costs on incoming edges), the graph G' given in Fig. 4 is built. Running the modified Dijkstra algorithm gives the spanning tree in Fig. 6, from which it can be seen that the max-bottleneck $s - T$ path is P': s, u, t, T with $B(P')=3$. An MRE $s - t$ path is therefore P: s, u, t with $D(P)=3$.

B. Modified MREP Algorithm (M-MREP)

In this section, we show how to find the optimal $s - t$ MRE path when the reverse energy cost $r_{i,j} \neq 0$.

First, we need to point out that a Dijkstra-like algorithm does not seem to be able to find the optimal path in such a case. The major reason why Dijkstra's algorithm works in general, both for the shortest path and the max-bottleneck problem, is because those problems possess the '*sub-solution optimality property*'. More specifically, that means if P: $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$ is an optimal $s - t$ path, P'': $s = v_1, v_2, \ldots, v_{k-1}$ is an optimal $s - v_{k-1}$ path. This property allows the Dijkstra's algorithm to find optimal paths correctly and quickly by building a spanning tree rooted at s in which every path from s to a node u is an optimal $s - u$ path.

However, in this problem with reverse energy cost, the 'sub-solution optimality property' does not always hold. In Fig. 7, an example is given. The unique $s - t$ MRE path is s, v, w, t, which has MRE 1. But s, v, w is not an $s - w$ optimal MRE path. Instead, it should be s, u, w, with MRE 2. Therefore, when finding an optimal $s - t$ path, it is not enough to just know all of the optimal $s - w$ paths for $(w, t) \in L$. For all $(w, t) \in L$,

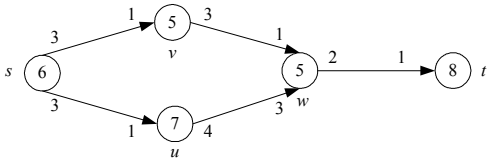

Fig. 7. An example for $r_{i,j} \neq 0$, in which the MRE optimal paths do not possess the "sub-solution optimality property".

we must know the optimal $s - w$ path whose final edge is (u, w), where $(u, w) \in L$. This observation will be the basis of our heuristic.

Our heuristic is to construct an augmented graph whose vertices V' are essentially the edges L of the original graph G. Given $G = (V, L)$, a new graph $G' = (V', L')$ is created. The vertices of G' are the edges of G plus two new edges (S, s) and (t, T), i.e.,

$$V' = \{(u,v) : (u,v) \in L\} \cup \{(S,s),(t,T)\}\} \quad (7)$$

The edges in L' are defined by

$$L' = \{((u,v),(v,w)) : (u,v),(v,w) \in L\} \quad (8)$$

Note that there is a one-to-one correlation between the $(S, s) - (t, T)$ paths $(S, s = v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k), (v_k = t, T)$ in G' and the $s - t$ paths $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$ in G. We now define the bandwidth $b()$ on the edges in G' as follows:

$$b((u,v),(v,w)) = \begin{cases} E(s) - s_{v,w} & \text{if } u = S \text{ and } v = s \\ E(t) - r_{u,t} & \text{if } v = t \text{ and } w = T \\ E(v) - r_{u,v} - s_{v,w} & \text{otherwise} \end{cases} \quad (9)$$

For path P in G, let P' denote the corresponding path in G'. Recall that for path P: $s = v_1, v_2, \ldots, v_{k-1}, v_k = t$ in G,

$$D(P) = \min \begin{pmatrix} E(v_1) - s_{v1,v2}, \\ E(v_k) - r_{v_{k-1},v_k}, \\ \min\limits_{1<i<k} \left(E(v_i) - r_{v_{i-1},v_i} - s_{v_i,v_{i+1}} \right) \end{pmatrix} \quad (10)$$

The important observation is that, if P is a *simple* path in G then, by definition of the bottleneck cost in G'

$$B(P') = D(P) \quad (11)$$

For example, the graph G' in Fig. 8 corresponds to the graph in Fig. 1. The simple $s - t$ path P: s, u, t with $D(P) = 2$ corresponds to the $(S, s) - (t, T)$ path P': (S, s), (s, u), (u, t), (t, T) which has the bottleneck cost $B(P')=2$.

Unfortunately, unlike in the previous case, finding a $(S, s) - (t, T)$ max-bottleneck path in G' does not always yields an optimal MRE path in G, since the optimal max-bottleneck path found might not be a simple path, i.e., it might include loops. The worse thing is that, even if the cycles are peeled away in the path to make it simple, the remaining path still might not be an optimal MRE path.

In order to avoid the loops from occurring in the routing process, we added some additional information, called the *associated set*. An associated set is basically the set of all the nodes in the original graph G along the path to a certain node in the transformed graph G'. For every step in the Dijkstra-like algorithm, before a node is inserted into the fringe set, its

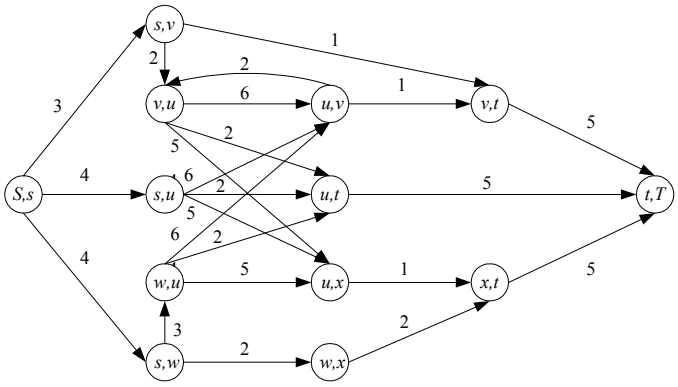

Fig. 8. The transformed graph G' corresponding to the graph G in fig. 1.

Modified MREP algorithm
(for finding MRE paths when $r_{i,j} \neq 0$)

Input: A graph G', Source (S, s) and destination (t, T).
Output: An MREP spanning tree rooted at (S, s) containing a path to (t, T).

1. **for** every node $(u, v) \in G'$ **do**
   $\{P[u, v] = 0; C[u, v] = -\infty\}$
2. $C[S, s] = +\infty$; $F = \phi$; $A[S, s] = \{s\}$
3. **for** every neighbor (s, w) of (S, s) **do**
   $\{P[s, w] = (S, s)$; $C[s, w] = b((S, s),(s, w))$; add (s, w) to $F\}$
4. select the node with maximum $C[u, v]$ from F;
   remove it from F and add it to the spanning tree
   set $A[u, v] = A[P[u, v]] \cap \{v\}$
5. **for** every neighbor (v, w) of (u, v)**do**
   if ($E(v) - s_{v,w} - r_{u,v} \geq 0$ and $w \notin A[u, v]$)
   {**if** ($C[v, w] = -\infty$)
   $\{P[v, w] = (u, v)$; $C[v, w] = \min(C[u, v], b((u, v),(v, w)))$; add (v, w) to $F\}$
   else if $\{(v, w) \in F$ and $C[v, w] < \min(C[u, v], b((u, v), (v, w)))$
   $\{P[v, w] = (u, v)$; $C[v, w] = \min(C[u, v], b((u, v), (v, w)))\}$
   }
6. **if** (F becomes empty) **return** failure
7. **repeat** step 4~6 **until** $C[t, T] \neq -\infty$ and $[t, T] \notin F$.

Fig. 9. The Modified MREP (M-MREP) algorithm

associated set needs to be checked. We call the algorithm based on the heuristic the *Modified MREP* (*M-MREP*) routing algorithm. The pseudo code for the M-MREP is given in Fig. 9.

In Fig. 9, the associated set is denoted as $A[\ ]$. Note in step 5, before a node is added to the fringe set, we have to make sure that its estimated residual energy after sending the message must not be negative. This is done to guarantee that the selected node has sufficient energy reservation to finish the current transmission task. Meanwhile, the associated set of the node is also checked. Simulation showed that this associated set checking procedure effectively avoids loops. Thus, we solved the non-simple path problem in M-MREP algorithm. Again, we attained the one-one correspondences just as the original MREP algorithm.

IV. PERFORMANCE EVALUATION

In this section, random graphs are presented in order to evaluate the performances of the proposed algorithm. We

generated 50 nodes randomly distributed in a square of size x by x. We tried 35 m (dense graph) and 50 m (sparse graph) for the value of x. Assume that the transmission range of each node is 10 m. The energy expenditure for sending one bit is assumed to be given by:

$$e = 0.1 \times d^2 (nJ / bit), \qquad d \le 10m \qquad (12)$$

where d is the distance in terms of meters. For the initial node energy, a value around 0.5 J (i.e. 500,000,000 nJ) looks more practical, but simulations showed that less initial energy does not affect the performance comparison. So we used lower levels for the initial energy. We assume that the packet size is fixed to 600 bits and the acknowledgement packet length is defined to be 120 bits. Each arrival corresponded to multiple fixed-size packets. Traffic was uniformly distributed between all source – destination pairs. All the results were based on ten randomly generated sample graphs.

A. Performance Measurement

In the following, three routing algorithms are compared: the Minimum Transmission Energy (MTE) routing algorithm that intends to minimize the total amount of energy for sending a packet, plus the two MREP algorithms (original MREP and M-MREP). The performance measure is the *system life*, defined as the time until the first node in the network runs out of energy. Since we did simulation in an event-driven fashion, we use the number of packets routed to denote the system life. Hence, the inter-arrival times of randomly generated traffic could be any distribution. Note that the exact number is not important. It is the difference between the results that indicate the performance of each routing algorithm.

Recall that finding the best MREP with backward energy is proven NP-hard. Therefore, for M-MREP, the system is dead when the algorithm cannot find a legal path. For the original MREP, the path can be found quickly. But it does not guarantee the complete transmission of a packet because it does not consider the backward energy cost. So the system is dead when we find transmitting a packet leads to the total drainage of a battery.

B. Results and Comparison

Because the performance depends on the network topology, we simulated two types of topologies: dense and sparse topologies. In the sparse network situation, as depicted in Fig. 10, we set the initial energy to 5,000,000 nJ. We can see that although MTE performs the worst, the difference is not very large. The "remaining energy level" is the average energy level of all the nodes, portrayed as percentages, after the system dies. Note that all the remaining energy levels are above 60%. Such high remaining energy levels mean that many other nodes still have energy even when one node has already used up all its battery energy. Unfortunately, this is typical of sparse networks because there are fewer choices when selecting paths for packets.

In dense topologies, we set the initial energy to 1,000,000 nJ. As shown in Fig. 11, MTE now performs far worse than MREP and MREP, while M-MREP performs a little bit better then its original counterpart MREP, but the improvement is marginal. However, if we increase the ACK packet length from 120 bits to 240 bits, M-MREP shows greater improvement.

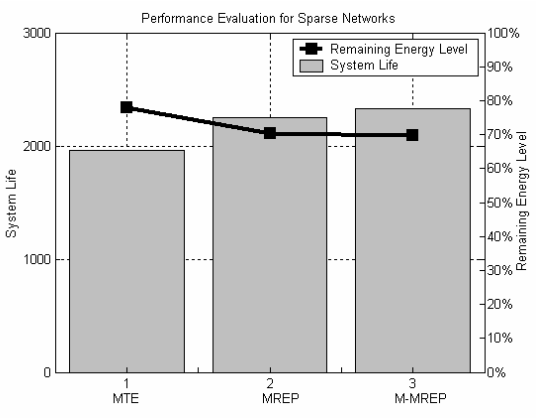

Fig. 10. Performance evaluation for sparse network

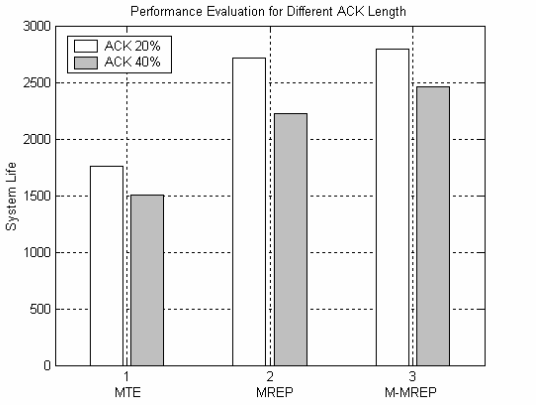

Fig. 11. Performance for different ACK length

MREP does not take account of the reverse energy cost, i.e., the energy to send an acknowledgement packet, but M-MREP does. If the reverse cost grows bigger, ignoring it will cause larger error. So we see the improvement of M-MREP is greater when acknowledgement packet is longer. But meanwhile, the system life will be reduced. Since the nodes have to spend more energy sending acknowledgement packets.

Fig. 12 shows the comparison between packet-by-packet routing and batch job routing (bulky arrival). The number of packets in each batch job is randomly generated using a geometric distribution. In our simulation, we used two different values for the mean value of the geometric distribution: 6 and 12. All the packets belong to one batch job were routed along the same path. It can be seen that all the algorithms perform less well as the number of packets in each batch increases. This is because a longer batch job means that the traffic is more difficult to distribute evenly throughout the whole network. This reduces the system life. The effect is more obvious for M-MREP. It seems that our heuristic algorithm is somewhat weak for the batch job routing.

We also found a problem with MREP: it does not consider total energy consumption at all. Consequently, sometimes the

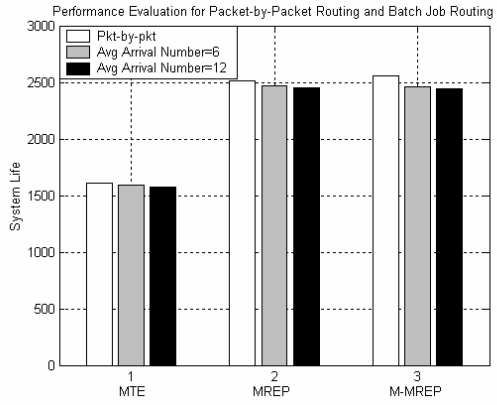

Fig. 12. Performance for packet-by-packet routing and batch job routing

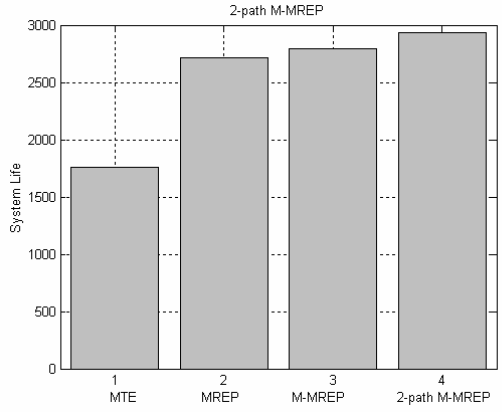

Fig. 13. 2-path M-MREP

MREP routing algorithm tends to find very long paths that, although better in terms of residual energy, are very wasteful in terms of energy consumptions. We developed the 2-path M-MREP. The essential idea is to find two paths (the best and the second best) in each path computation. We then compare the total energy consumed by the two paths. If the best MRE path consumes more energy than the second path by a fixed amount (set at 7% in our simulations), then the second best path is chosen. Simulation showed that this method can improve the system life (Fig. 13). This method works for the original MREP, too. We also tried 3-path algorithm, but its improvement over the 2-path algorithm is marginal.

V. CONCLUSION

In this paper, we introduced the idea of finding an Maximum Residual Energy Path (MREP) in ad hoc networks. Routing in such kind of network is quite different from the one in Internet. Here, battery energy at network nodes is a very limited resource and need to be utilized efficiently. So to extend system life is our objective when doing routing. The MREP algorithm works pretty well, but it doesn't consider the reverse energy cost. It has been proved that finding MRE path when considering reverse energy is *NP-complete*.

We showed a heuristic algorithm – Modified Maximum Residual Energy Path (M-MREP) algorithm, which guarantees that when a path is found, it will have enough energy to transmit the packet, while the original MREP cannot guarantee that. We also presented a routing technique that can improve all MREP-related routing schemes. These techniques will be useful for energy-conservation routing in ad hoc networks.

REFERENCES

[1] J. C. Haartsen, S. Mattisson, "Bluetooth – A New Low-Power Radio Interface Providing Short-Range Connectivity", *IEEE Proceedings*, vol. 88, no. 10, pp. 1651-1661, October 2000

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", *Computer Networks*, vol 38, issue 4, pp. 393-422, March 2002.

[3] J.-H. Chang, L. Tassiulas, "Energy Conserving Routing in Wireless Ad-hoc Networks", in *Proceeding of IEEE Infocom*, 2000, Israel.

[4] J.-H. Chang, L. Tasssiulas, "Routing for maximum system lifetime in wireless ad-hoc networks", in *Proceedings of 37-th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Sept. 1999.

[5] I. Stojmenovic, X. Lin, "Power-aware localized routing in wireless networks" in *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, vol. 12, 2001

[6] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pp.3005-3014, Hawaii, January 2000

[7] S. Singh, M. Woo, C. S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks", in *The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 181-190, 1998

[8] Rudolf Fleischer, Mordecai J. Golin, Chin-Tau Lea, and Steven Wong, "Finding optimal path in MREP routing" in preparation.

[9] N. Malpani, J. Chen, "A Note on Practical Constructions of Maximum Bandwidth Paths", http://www.cs.tamu.edu/course-info/cpsc629/chen/notes/

[10] J. Edmonds, R. M. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network flow problem", *Journal of the ACM*, 19(2), pp. 248-164, April 1972.

[11] C. J. Perkins, "Ad Hoc Networking", Addison- Wesley Pub. Co., 1st edition, 2000