Non-Blocking, Localized Routing Algorithm for Balanced Energy Consumption in Mobile Ad Hoc Networks[1]

Kyungtae Woo and Chansu Yu
School of Engineering
Information and Communications University
58-4 Hwaam-Dong, Yusung-Ku
Taejon 305-372, Korea
{hall00, cyu}@ icu.ac.kr

Hee Yong Youn
School of Electrical and Computer Engineering
Sungkyunkwan University
300 Chunchun-Dong, Jangang-Ku
Suwon 440-746, Korea
youn@ece.skku.ac.kr

Ben Lee
Department of Electrical and Computer Engineering
Oregon State University
ECE Bldg. 302
Corvallis, OR 97331
benl@ece.orst.edu

Abstract

As mobile computing requires more computation as well as communication activities, energy efficiency becomes the most critical issue for battery-operated mobile devices. Specifically, in ad hoc networks where each node is responsible for forwarding neighbor nodes' data packets, care has to be taken not only to reduce the overall energy consumption of all relevant nodes but also to balance individual battery levels. Unbalanced energy usage will result in earlier node failure in overloaded nodes, and in turn may lead to network partitioning and reduced network lifetime. There has been active research on developing energy-aware routing protocols for mobile ad hoc networks. They use power-related metric, such as minimizing energy consumed per packet, and attempt to find an optimal route using global information. Even though these algorithms save energy and maximize the system life, they have limited practical value because they require global information of all relevant nodes in order to compare and choose the best route. This paper presents a new routing algorithm, called *Local Energy-Aware Routing (LEAR)*, which achieves a trade-off between balanced energy consumption and shortest routing delay, and at the same time avoids the blocking and route cache problems. Our performance study based on *GloMoSim* simulator shows that LEAR improves the energy balance 10-35% depending on node mobility.

Keywords: Mobile ad hoc networks, energy consumption, source routing, wireless communication.

1 Introduction

Mobile devices coupled with wireless network interfaces are likely to become a pervasive part of future computing infrastructures with technical advancements in *wireless communication, mobility* and *portability* [1]. Among them, portability may be the most critical issue in these battery-operated devices since battery imposes power, weight and size constraints. In order to provide improved portability, it is imperative to use *low-power components* and *energy-efficient operations*. As the trend in mobile computing is towards more communication-dependent activities and energy consumption due to the wireless communication can represent more than half of total system power [2], the key to energy efficiency is at the energy-aware network protocols such as links, MAC, routing, and transport protocols.

This paper addresses the issue of energy-conserving routing protocols in *ad hoc networks* of mobile hosts. Ad hoc networks are multi-hop, wireless networks where all mobile hosts or nodes cooperatively maintain network connectivity without communication infrastructures for routing. When communicating mobile nodes are not within their radio range, a data packet "*hops*" through intermediate nodes to reach the

[1] This research was supported in part by the Ministry of Information and Communication under Grant No. 99-159-01.

destination. Thus, each mobile node operates not only as a host but also as a router to forward data packets on behalf of other nodes. As communicating as well as intermediate nodes move around, the routing protocol must adapt its routing decision to enable continued communications between the nodes. Many different routing protocols have been proposed in the literature [3-9] and submitted to the *Internet Engineering Task Force (IETF) Mobile Ad Hoc Network (MANET) Group*. A major issue in these algorithms is to find a shortest path consisting of minimum number of intermediate forwarding nodes between a source and a destination. However, it is possible that some particular mobile nodes are unfairly burdened to support many packet-relaying functions. This *hot spot* node may consume more battery energy and stops running earlier than other nodes disrupting the overall ad hoc network. This is particularly true for some optimized routing protocols that prefer specific mobile nodes in the selection of routing paths. For example, slow-moving nodes can be regarded as better candidates since they are likely to forward packets for a longer duration than fast-moving nodes. However, this may adversely affect the distribution of energy consumption among all mobile nodes.

Recently, there has been active research in this regard to improve the energy efficiency of the wireless communication subsystem. *Cano and Manzoni* evaluated several ad hoc routing algorithms in terms of energy consumption [10]. *Singh et al.* proposed power-aware routing algorithms that try to find a low-energy route instead of the shortest routing path [11]. They suggested several energy-related metrics: *minimizing energy consumed/packet*, *maximizing time to network partition*, *minimizing variance in node power levels*, *minimizing cost/packet*, or *minimizing maximum node cost*. The first metric is useful to minimize the overall energy consumption of all nodes. But, as they pointed out, the routing algorithm based on the first metric tends to select shortest paths and thus it does not achieve balanced energy consumption among the nodes. Other metrics can be used to maximize the network lifetime by avoiding the hot spot problem [11]. Two other routing algorithms in this direction can be found in [12,13], where the latter achieves the goal by controlling the transmit power of the communication device as suggested in [14].

However, these studies have three major shortcomings. First, they assume a static network topology where mobile nodes do not move, and thus ad hoc routing algorithms to support mobility are not considered. This assumption simplifies their studies but the validity of their results is limited. Second, their algorithms are *blocking* in the sense that a node must wait until all possible paths have been evaluated according to the given power-related metric to select the best routing path. This is necessary since the algorithms require global information of all relevant nodes along every possible path. For this reason, we call the algorithms *Global Energy-Aware Routing (GEAR)*. Third, their simulation studies do not consider the *route cache*, which is an important optimization technique used in most ad hoc routing protocols [3].

In this paper, we propose a new ad hoc routing protocol, called *Local Energy-Aware Routing (LEAR)*, which achieves a balanced energy consumption among all participating mobile nodes. When a routing path is searched for, each mobile node relies on local information of *remaining battery level* to decide whether to participate in the selection process of a routing path or not. An energy-hungry node can conserve its battery power by not forwarding data packets on behalf of others. Decision-making process in LEAR is distributed to all relevant nodes, and the destination node does not need wait or *block* itself. The proposed scheme efficiently utilizes the route cache and avoids the blocking problem since it does not require global information. To the best of our knowledge, this is the first work to explore the balanced energy consumption in a realistic environment where routing algorithms, mobility, and radio propagation models are all considered. We implemented the LEAR protocol based on *Dynamic Source Routing (DSR)* [3], which is a simple but efficient ad hoc routing algorithm. The performance of LEAR is evaluated based on *GloMoSim 2.0* simulator [15] developed at *UCLA*. We measure the energy consumptions of the mobile nodes and obtain the *standard deviation* as well as *peak-to-mean ratio* to estimate the distribution. Simulation results show that compared to DSR the proposed LEAR algorithm improves the energy balance as much as 35% when node mobility is high and 10% when it is low.

This paper is organized as follows. Section 2 overviews several ad hoc routing algorithms including DSR. In Section 3, we propose an energy-aware routing algorithm LEAR. Performance evaluation of the proposed scheme is presented in Section 4. Section 5 discusses the related work on other energy-efficient network protocols than the routing layer. Finally, concluding remarks are found in Section 6.

2 Ad Hoc Routing Algorithms

Routing protocols for mobile ad hoc networks generally fall into one of two categories: *proactive* or *reactive* [16]. Proactive routing protocols are derived from well-known distributed adaptive routing schemes for fixed networks, such as *distance-vector* or *link state algorithms*. These methods attempt to maintain routes to all destinations at all times, regardless of whether they are needed. They react to changes in the network topology by broadcasting updates throughout the network to maintain a consistent network view. Information updates can be topology-driven, periodic, or both. On the other hand, reactive, or so-called *on-demand*, routing algorithms find a route only when desired by a source node. Under highly dynamic link conditions, reactive protocols are expected to generate fewer overhead messages and provide a more reliable routing than proactive routing protocols. See [16,17] for a general overview of these protocols as well as extensive performance comparisons, and [18] for scenario-based performance analysis.

Dynamic Source Routing (DSR) protocol [3] is an on-demand routing algorithm based on the concept of source routing. We adopt it as our baseline routing algorithm throughout this paper because of its simplicity and efficiency. Two main components of DSR are *source route* and *route cache*, and two main steps of DSR are *route discovery procedure* and *route maintenance procedure*. When a mobile node has a packet to send to a destination, it initiates the route discovery procedure by broadcasting a *route request message (ROUTE_REQ)*. Intermediate nodes piggyback their identities on the source route included in *ROUTE_REQ* message and broadcast again[2]. Each node, whether it is the sender, the destination, or an intermediate node, receives multiple messages along different paths but chooses the best one according to the path length. Since the first arriving message usually contains the shortest source route, it is reasonable to choose this message and ignore all others[3]. In that sense, the DSR protocol is said to be *non-blocking* because every

[2] A node forwards the request message but it also sends it back to the original sender because the message is transmitted to all one's neighbors in a mobile network. In the DSR protocol, a mobile node discards a route request message if it recently saw the same request or if it's identity is already included in the source route [3].

[3] One possible optimization in actual implementation is that the destination node sends a reply message for each arrived request message. These reply messages indicate the path to the destination to intermediate nodes not only along the shortest path but also along all other paths found. However, it is nothing to do with the selection of the shortest path. For this reason, we do not include those details in describing routing algorithms in this paper.

participating mobile node does not have to wait indefinitely for more messages to arrive after receiving the first one. A destination node simply reacts to a *ROUTE_REQ* message by immediately sending a *route reply message (ROUTE_REPLY)* to the source and ignores all later messages having the same source-destination pair.

However, the *route discovery procedure* tends to cause a traffic surge as the query is propagated through the network. *Route cache*, which is another main component of DSR, is used to reduce traffic. Each mobile node maintains its own route cache that contains the source routes destined for other mobile nodes. Entries in the route cache are continually updated as new routes are learned. Figure 1 shows the optimization technique based on the route cache. In this example, a source node (S) initiates a route discovery procedure by broadcasting *ROUTE_REQ* message to find a path to a destination node (D). Intermediate nodes (for example, node A) forward the message. However, node B stops flooding since an available route to the destination (D) is found in its route cache. A *ROUTE_REPLY* message is then generated, which includes the identities of intermediate nodes recorded in the route cache (C_1 and C_2 in this example).

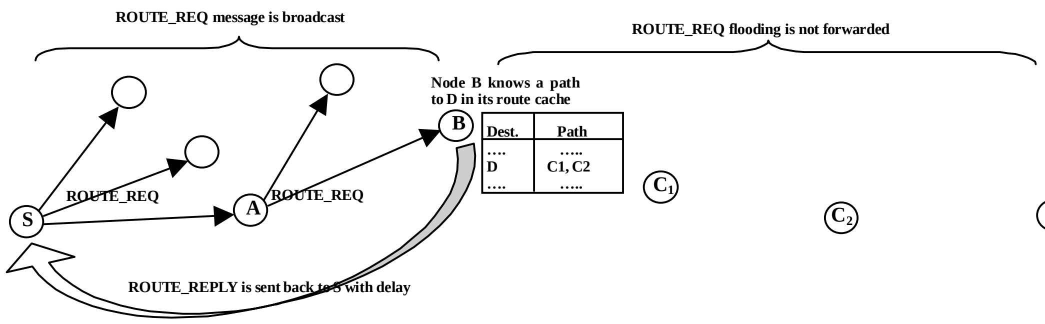

ROUTE_REQ message is broadcast
ROUTE_REQ flooding is not forwarded
Node B knows a path to D in its route cache
B
Dest.
Path
….
…..
D
C1, C2
….
…..
C_1
C_2
D
S
A
ROUTE_REQ
ROUTE_REQ
ROUTE_REPLY is sent back to S with delay

Figure 1. Route Cache in Dynamic Source Routing (DSR)

Table 1. Dynamic Source Routing (DSR) Algorithm with Non-Blocking Property

Node	Steps
Source node	***Broadcast a route request message (ROUTE_REQ);*** ***Wait for the first arriving route reply message (ROUTE_REPLY);*** ***Select the source route contained in the message;*** ***Ignore all later replies;***
Intermediate node	***Upon receipt a ROUTE_REQ,*** ***If it has the route to the destination in its route cache, delay for a deterministic duration of time and send a route reply message to the source;*** ***Otherwise, forward (broadcast) a ROUTE_REQ;*** ***Ignore all later requests;***
Destination node	***Upon receipt of the first arriving ROUTE_REQ, send a ROUTE_REPLY to the source with the source route contained in the message;*** ***Ignore all later requests (see footnote 3 above);***

Replies from intermediate nodes may generate *packet collisions* and *local congestion* due to the route cache optimizations, and the source may not able to find out the real shortest path. Therefore, it has been suggested that an intermediate node introduce a delay of $H \times (h\text{-}1+r)$ before replying (as if it travels around from the destination), where H is a small constant delay introduced per hop, h is the length in number of network hops for the route to be returned in this host's reply, and r is a random number between 0 and 1 [3]. This way an intermediate node needs to wait only for a deterministic time and thus it does not affect the non-blocking property of the DSR. Table 1 summarizes the DSR algorithm.

3 Energy-Balancing Ad Hoc Routing Algorithms

In this section, we introduce power-aware routing algorithms. A straightforward approach suggested in [11] is to use a power-related metric and try to find an optimal energy-conserving route based on global information. Subsection 3.1 discusses an implementation of the approach based on DSR algorithm. We call the power-aware routing algorithm *Global Energy-Aware Routing (GEAR)*. Even though GEAR can save energy and maximize the system lifetime, we show that it has two major disadvantages that can be avoided with *Local Energy-Aware Routing (LEAR)* protocol as discussed in Subsection 3.2.

3.1 Global Energy-Aware Routing (GEAR) Algorithm

As explained in the previous section, *ROUTE_REQ* is propagated towards the destination node in the original DSR algorithm. In GEAR, each node piggybacks its power-related measure (such as the remaining battery power) as well as its identity on the *ROUTE_REQ* message and forwards (broadcasts) it. The destination node receives multiple request messages but chooses the best route with respect to the given power metric. Table 2 summarizes the GEAR algorithm.

Table 2. Global Energy-Aware Routing (GEAR) Algorithm

Node	Steps
Source node	***Broadcast a ROUTE_REQ;*** ***Wait for the first arriving ROUTE_REPLY;*** ***Select the source route contained in the message;*** ***Ignore all later replies;***
Intermediate node	***Forward (broadcast) the ROUTE_REQ;*** ***Ignore all later requests;***
Destination node	***Upon receipt the first arriving ROUTE_REQ, wait for all later ROUTE_REQs to arrive with the same source-destination pair;*** ***Select the route which minimizes the power metric;*** ***Send a ROUTE_REPLY to the source with the source route contained in the message;***

Two major disadvantages of GEAR are difficulty in utilizing the route cache and the blocking property. GEAR protocol inherently cannot utilize the route cache because a node does not have power-related information of the following nodes recorded in its route cache. Without the route cache, we expect traffic surge due to the flood of route request messages as explained in Section 2. Another difficulty is in estimating how long the destination node has to wait before selecting the best route. In order to compare and choose the best one, it has to wait until it receives all request messages along all possible routing paths. Some time duration can be specified, but if it is too short, some routing paths with a better metric may not be considered. On the other hand, long duration will affect the average response time.

3.2 Local Energy-Aware Routing (LEAR) Algorithm

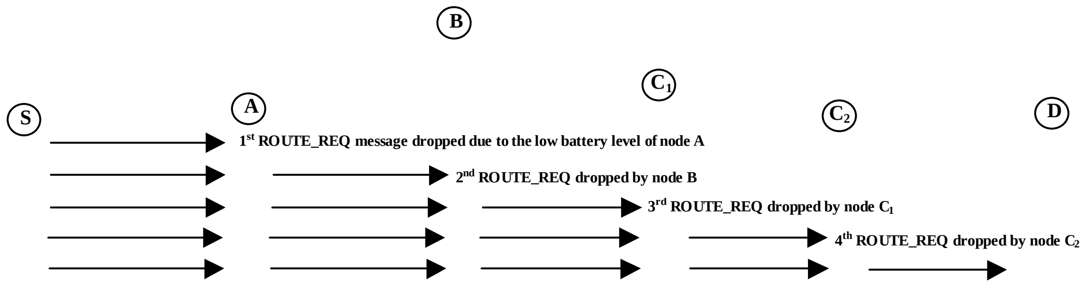
Localized, power-aware routing algorithm based on DSR is now described. In the original DSR algorithm, each mobile node has no choice but to join in on the route selection process. The basic idea of the LEAR protocol is to consider each mobile node's willingness to participate in the routing path and to forward data packets on behalf of others. Each node determines whether to accept and forward the *ROUTE_REQ* message or not depending on its *remaining battery power* (E_r). When it is higher than a *threshold value* (Th_r), the *ROUTE_REQ* is forwarded; otherwise, the message is dropped. The destination will receive a route request message only when all intermediate nodes along the route have good battery levels. Thus, the first message to arrive is considered to follow an energy-efficient as well as reasonably short path. In contrast, GEAR optimizes energy while the original DSR optimizes delay, but not both. In addition, LEAR is *non-blocking* since the destination node can immediately respond to the first arriving request message.

When any one intermediate node has lower battery level than its threshold value ($E_r < Th_r$), a *ROUTE_REQ* is simply dropped. If this occurs for every possible path, the source will not receive a single reply message even though there exists a path between the source and the destination. To prevent this, the source will re-send the same route request message, but this time with an increased *sequence number*. When an intermediate node receives the same request message again with a larger sequence number, it adjusts (lowers) its Th_r by d to allow forwarding to continue. Table 3 describes the basic operation behavior of the LEAR algorithm.

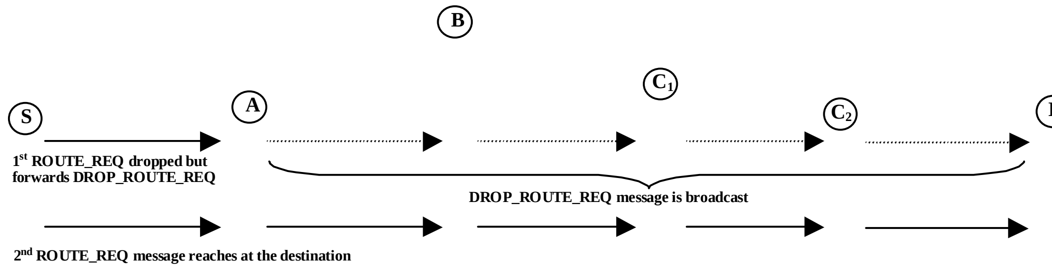
Table 3. Local Energy-Aware Routing (LEAR) Algorithm

Node	Steps
Source node	***Broadcast a ROUTE_REQ;*** ***Wait for the first arriving ROUTE_REPLY;*** ***Select the source route contained in the message;*** ***Ignore all later replies;***
Intermediate node	***If the message is not the first trial and*** $E_r < Th_r$, ***adjust (lower)*** Th_r ***by d;*** ***If*** $E_r > Th_r$, ***forward (broadcast) the ROUTE_REQ and ignore all later requests;*** ***Otherwise, drop the message;***
Destination node	***Upon receipt the first arriving ROUTE_REQ, send a ROUTE_REPLY to the source with the source route contained in the message;***

LEAR provides the shortest routing path among multiple energy-rich paths. However, in its basic form, it cannot utilize the route cache (as in GEAR) and it may incur repeated route request messages due to dropping of ROUTE_REQ messages. To alleviate these two problems, four additional routing-control messages are utilized: *DROP_ROUTE_REQ*, *ROUTE_CACHE*, *DROP_ROUTE_CACHE* and *CANCEL_ROUTE_CACHE*. Repeated request messages are caused by repeated discovery procedures as shown in Figure 2(a). For example, consider when a source node (S) sends a *ROUTE_REQ* message to find the path to a destination (D). If intermediate nodes A, B, C_1 and C_2 have lower E_r's than the required Th_r's, node A will drop the request message. S will resend the *ROUTE_REQ* message with an increased sequence number. Then, node A adjusts its threshold value and forwards the message, but this time node B will drop it. Destination node D will finally receive a *ROUTE_REQ* message at the fifth route discovery procedure. The control message, *DROP_ROUTE_REQ*, is used to avoid this cascading effect. When a node drops a *ROUTE_REQ* message, it instead forwards a *DROP_ROUTE_REQ* message. The subsequent nodes closer to the destination now know that a request message was dropped and lower their threshold values when they receive the second *ROUTE_REQ* message. As can be seen in Figure 2(b), the second route request message can now reach the destination.


B
C1
S
A
C2
D
1st ROUTE_REQ message dropped due to the low battery level of node A
2nd ROUTE_REQ dropped by node B
3rd ROUTE_REQ dropped by node C1
4th ROUTE_REQ dropped by node C2

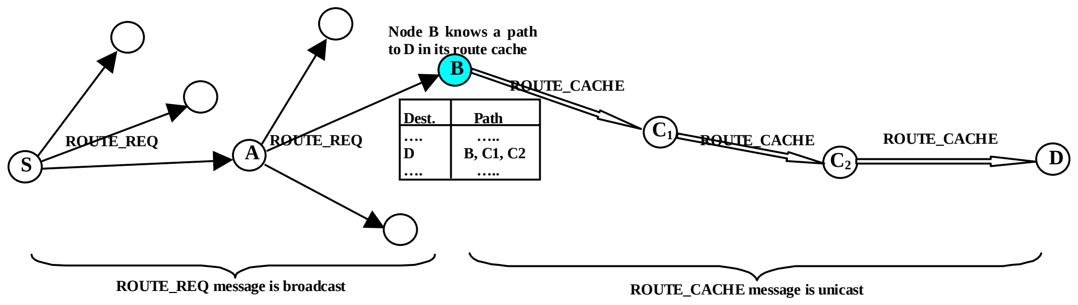
(a) Five route discovery procedures are required


B
C1
S
A
C2
D
1st ROUTE_REQ dropped but forwards DROP_ROUTE_REQ
DROP_ROUTE_REQ message is broadcast
2nd ROUTE_REQ message reaches at the destination

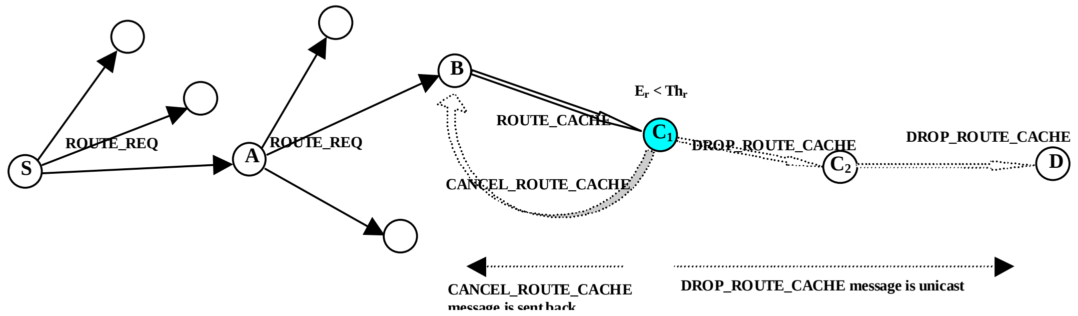
(b) Two route discovery procedures are required

Figure 2. Repeated Route Request Messages in LEAR

Another problem with the LEAR algorithm is that it does not exploit the optimization technique based on route cache. In the original DSR protocol, when an intermediate node receives a *ROUTE_REQ* and finds a route to the destination in its route cache, it stops broadcasting and replies to the source immediately. In LEAR, an intermediate node cannot reply because it does not have information on the battery levels of the nodes included in the cache entry (the following intermediate nodes from itself to the destination). For example shown in Figure 3(a), a source node (S) broadcasts a *ROUTE_REQ* message destined for node D. Even though node B knows the path to D from its route cache, it cannot stop forwarding and reply to S because the battery levels of nodes C_1 and C_2 are not known. Therefore, LEAR utilizes a special unicast message (*ROUTE_CACHE*) to determine the battery levels of the intermediate nodes along the path to the destination recorded in its route cache (see Figure 3(a)). This way, the route cache is properly utilized and at the same time the flooding of route request messages is avoided. Note that the destination node may receive multiple *ROUTE_REQ* messages and multiple *ROUTE_CACHE* messages, but chooses the first arriving one to reply to the sender.


Node B knows a path to D in its route cache
S
A
B
C_1
C_2
D
ROUTE_REQ
ROUTE_REQ
ROUTE_CACHE
ROUTE_CACHE
ROUTE_CACHE
Dest.
Path
….
…..
D
B, C1, C2
….
…..
ROUTE_REQ message is broadcast
ROUTE_CACHE message is unicast

(a) Unicast message to inform to the destination



(b) Invalidating route cache upon a node with low battery level

Figure 3. Exploiting Route Cache in LEAR

Complications arise when a node (node C_1 in Figure 3(b)) has lower battery level than its threshold value ($E_r < Th_r$) along the unicast of a *ROUTE_CACHE* message. As in the normal *ROUTE_REQ* message, it informs this situation to the subsequent nodes (nodes C_2 and D) by sending *DROP_ROUTE_CACHE* message in order for them to adjust their power levels by *d* at the next route request message. Another special message, *CANCEL_ROUTE_CACHE*, is sent back to the node that started sending the *ROUTE_CACHE* message (node B) so that it can invalidate the entry in its route cache. This is required since there may be a more energy-efficient route from node B, but other paths will never be explored as long as it has an entry to node D in its route cache. Table 4 describes the complete LEAR algorithm.

Table 4. Complete LEAR Algorithm

Node	Steps
Source node	***Broadcast a ROUTE_REQ;*** ***Wait for the first arriving ROUTE_REPLY;*** ***Select the source route contained in the message;*** ***Ignore all later replies;***
Intermediate node	***Upon receipt a ROUTE_REQ,*** ***If the message is not the first trial and $E_r < Th_r$, adjust (lower) Th_r by d;*** ***If it has the route to the destination in its cache,*** ***if $E_r > Th_r$, forward (unicast) ROUTE_CACHE and ignore all later requests;*** ***otherwise, forward (unicast) DROP_ROUTE_CACHE and ignore all later requests;*** ***Else,*** ***if $E_r > Th_r$, forward (broadcast) ROUTE_REQ and ignore all later requests;*** ***otherwise, forward (unicast) DROP_ROUTE_REQ and ignore all later requests;***
	Upon receipt a ROUTE_CACHE, ***If the message is not the first trial and $E_r < Th_r$, adjust (lower) Th_r by d;*** ***If $E_r > Th_r$, forward (broadcast) ROUTE_CACHE and ignore all later requests;*** ***Otherwise, forward (unicast) DROP_ROUTE_REQ and ignore all later requests;*** ***and send backward (unicast) CANCEL_ROUTE_CACHE;***
Destination node	***Upon receipt the first arriving ROUTE_REQ or ROUTE_CACHE, send a ROUTE_REPLY to the source with the source route contained in the message;***

4 Performance Evaluation

This section provides the simulation results and shows the effectiveness of the proposed LEAR algorithm compared to the original DSR. We used *GloMoSim 2.0* simulator [15], which is a scalable simulation environment for wireless and wired networks based on *Parsec* [19]. GloMoSim supports a wide range of ad hoc routing protocols as well as realistic physical layers. Table 5 shows the parameters used for the simulation study.

Our evaluations are based on the simulation of 40 mobile nodes (*NUMBER-OF-NODES*) moving about over a regular rectangular area of 1000 meters by 1000 meters (*TERRAIN-RANGE-X* and *TERRAIN-RANGE-Y*) for 500 seconds of simulated time (*SIMULATION-TIME*). In order to provide a fair comparison, the same set of seed numbers was used for different routing algorithms (*SEED*). The seed is used to generate random numbers for selecting initial positions of the mobile nodes (*NODE-PLACEMENT*) as well as for specifying mobility pattern (*MOBILITY*). We assume that mobile nodes move randomly according to the "*random waypoint*" model [1]. Each node begins the simulation by staying at the initial position for a predefined pause time (*MOBILITY-WP-PAUSE*). It then selects a random target position in the simulated area and moves to that direction at a randomly chosen speed between two parameters (from *MOBILITY-WP-MIN-SPEED* to *MOBILITY-WP-MAX-SPEED* or 0~5 meters/second). The node repeats this mobility behavior after reaching the target.

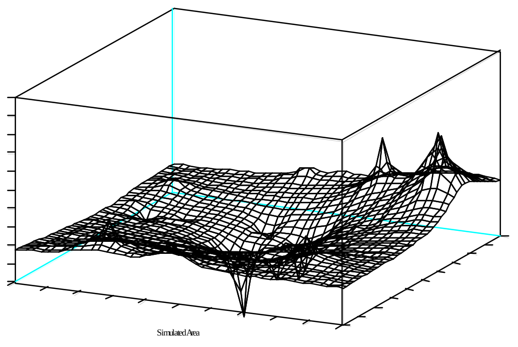
Table 5. Input Configuration File for GloMoSim Simulator

```
# Terrain Area we are simulating: 1000 meters × 1000 meters
TERRAIN-RANGE-X 1000
TERRAIN-RANGE-Y 1000

# Power range of wireless nodes in the simulation: 250 meters
POWER-RANGE 250

# Random number seed
SEED 1

# Maximum simulation time: 500 seconds
SIMULATION-TIME 500S

# Number of nodes being simulated: 40 nodes
NUMBER-OF-NODES 40

# Node placement strategy: initial position of 40 nodes are
# randomly selected
NODE-PLACEMENT RANDOM
#NODE-PLACEMENT UNIFORM
#NODE-PLACEMENT GRID
#NODE-PLACEMENT FILE

# Radio propagation model: free space
PROPAGATION-FUNC FREE-SPACE
#PROPAGATION-FUNC RAYLEIGH
#PROPAGATION-FUNC RICEAN

# Bandwidth (in bits per second): 2Mbps
BANDWIDTH 2000000

# Radio layer: capture
RADIO-TYPE RADIO-CAPTURE
#RADIO-TYPE RADIO-NO-CAPTURE

# MAC layer: IEEE 802.11
MAC-PROTOCOL 802.11
#MAC-PROTOCOL CSMA
#MAC-PROTOCOL MACA
```

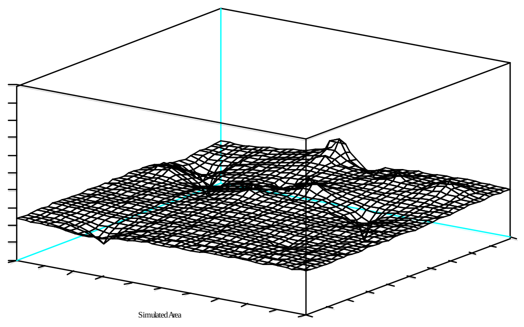
```
# Routing protocol: DSR
ROUTING-PROTOCOL DSR
#ROUTING-PROTOCOL BELLMANFORD
#ROUTING-PROTOCOL OSPF

#NETWORK-PROTOCOL IP
TRANSPORT-PROTOCOL-TCP YES
TRANSPORT-PROTOCOL-UDP YES

# Interested statistics: radio layer provides
#                        energy consumption values
TCP-STATISTICS NO
UDP-STATISTICS NO
ROUTING-STATISTICS NO
NETWORK-LAYER-STATISTICS NO
MAC-LAYER-STATISTICS NO
RADIO-LAYER-STATISTICS YES
CHANNEL-LAYER-STATISTICS NO

# Mobility: random-waypoint
#             speed is from 0 to 5 meters/sec
#             pause time is 50, 100, 150, 200, 250, 300, 350 or
#             400 seconds
MOBILITY RANDOM-WAYPOINT
MOBILITY-WP-PAUSE 50S
MOBILITY-WP-MIN-SPEED 0
MOBILITY-WP-MAX-SPEED 5
#MOBILITY NONE
#MOBILITY TRACE
#MOBILITY REFERENCE-POINT-GROUP
#MOBILITY BBN
#MOBILITY PATHLOSS-MATRIX
#MOBILITY RANDOM-DRUNKEN
MOBILITY-POSITION-GRANULARITY 0.5
```

A separate application configuration file specifies traffic as well as application type: *FTP*, *HTTP*, *Telnet* or *constant bit rate (CBR)*. In our simulation, five CBR sources and their corresponding destinations are randomly selected among 40 mobile nodes. Each CBR source sends five 1024-byte packets every second for a specified duration. Since the data transmission is based on UDP rather than TCP, some data packets can be lost. GloMoSim simulates a realistic physical layer that includes a *radio propagation model*, *radio network interfaces*, and the *IEEE 802.11 Medium Access Control (MAC)* protocol using the *Distributed Coordination Function (DCF)*. The *radio network interface card (NIC)* model includes collisions, propagation delay and signal attenuation with a 2Mbps (*BANDWIDTH*) data rate and a radio range of 250 meters (*POWER-RANGE*).

In this paper, we are specifically interested in energy consumption and its balance across all mobile nodes. For each node, energy consumption is measured at the radio layer during the simulation. According to the specification of *IEEE 802.11*-compliant *WaveLAN-II* [20] from Lucent, the power consumption varies from 0.045 Watts (9mA $\times$ 5 Volts) in sleep mode to 1.25~1.50 Watts (230~250mA $\times$ 5 Volts) in receiving and transmitting modes, respectively (see Section 5). The instantaneous power is multiplied by time delay to obtain the energy consumed. For example, data transmission of a 1024-byte packet consumes 6.14×10^{-3} Joules (1.50 Watts $\times$ 1024$\times$8 bits / 2000000 bps). Two energy-related assumptions were made for our simulation study. First, the energy consumption during idling was ignored. Since a node stays idle most of time[4], a general idea to conserve energy is to put the node in sleep mode while idling and make the node consume negligible energy. Second, *non-promiscuous receive mode* [3] was assumed. Since a node does not know when others send packets to itself, it should be in promiscuous receive mode. However, emerging standards for wireless LANs such as *IEEE 802.11* [21] and *Bluetooth* [22] provides a mechanism for each node to know when to wake up and receive packets and to sleep rest of the time. Thus, time delay due to data receive is similar to that due to data transmission for a relaying node. Without this assumption, energy consumption is dominated by data receive or *overhearing* [10] and the proposed LEAR algorithm may provide a limited benefit.



(a) With the original DSR algorithm



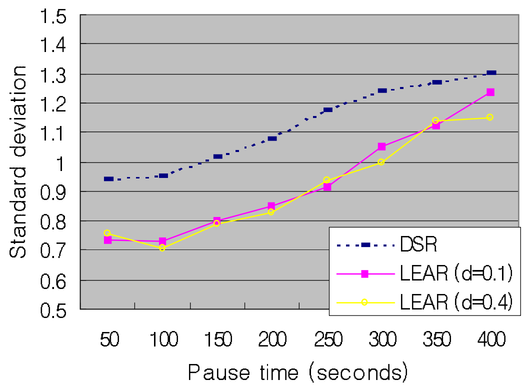
(b) With the LEAR algorithm

Figure 4. Contours of Remaining Battery Powers of 40 Mobile Nodes

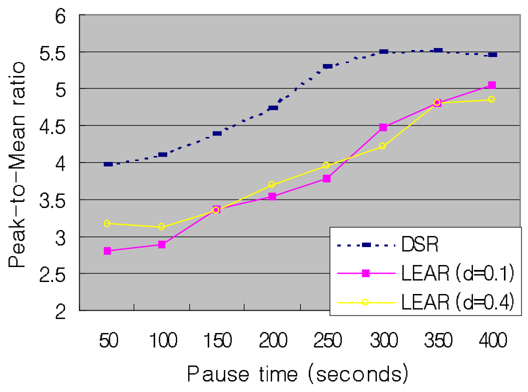
Figure 4 shows the effectiveness of the proposed LEAR algorithm compared to the original DSR.

[4] In this study, a CBR source transmits five 1024-byte data packets per second for 500 seconds of the simulation time, translating 10.24 seconds of transmission delay (1024 $\times$ 8 / 2000000 bps $\times$ 500 seconds $\times$ 5 packets/second). Some of the rest 489.76 seconds are used to receive data but most of time the node will do nothing.

Figures 4(a) and (b) depict the contour of remaining battery powers of 40 mobile nodes in 1000m × 1000m simulated area with DSR and LEAR, respectively. The graphs have been smoothed using simple interpolation. It can be easily seen from the figure that the proposed LEAR achieves balanced energy consumption across all mobile nodes. With the original DSR, some nodes consume less energy while others consume much more, which results in earlier death of those particular nodes leading to shorter network lifetime. For this simulation study response time was not considered. The proposed LEAR algorithm may result in longer transmission time compared to the DSR because it gives precedence to an energy-efficient route over the shortest path. However, the extra delay is negligible in a limited physical range where the ad hoc networking service is provided.

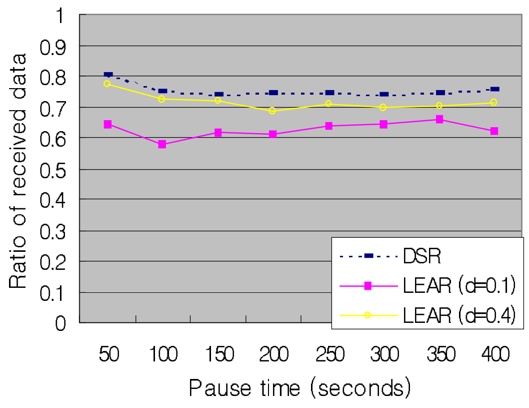


(a) Standard deviation

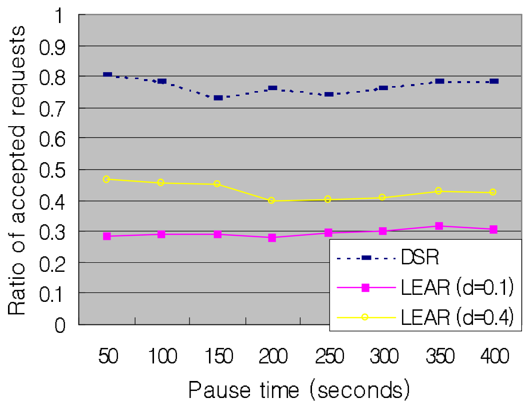


(b) Peak-to-Mean ratio

Figure 5. Distribution of Energy Consumptions of 40 Mobile Nodes



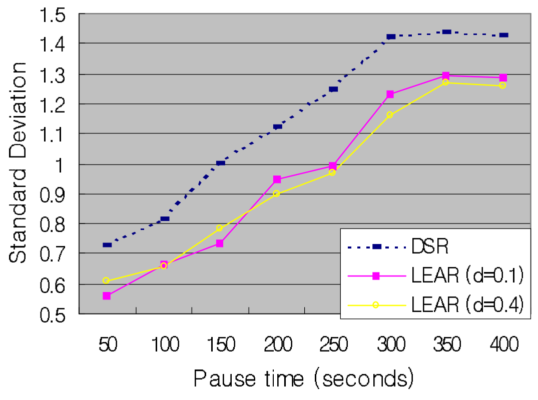
(a) Ratio of received data



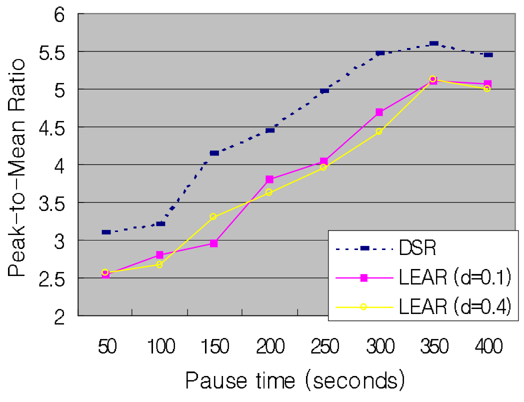
(b) Ratio of accepted*ROUTE_REQs*

Figure 6. Other Simulation Results

Detailed simulation results on the distribution of energy consumptions are presented in Figure 5. Each data shown in Figure 5 and the following figures are obtained by taking average of 100 simulation runs. We obtained *standard deviation* and *peak-to-mean ratio* as the performance metrics to measure the distribution. Pause time is varied from 50 to 400 seconds. We exclude the CBR source and destination nodes since they must transfer data packets and consume much larger energy than other nodes regardless of their remaining battery power. Initial threshold value (Th_r) of each mobile node is set to 90% of its initial battery energy and the adjustment value (d) is either 0.1 or 0.4. As shown in Figure 5, LEAR improves the energy balance as much as 35% when node mobility is high (50 seconds of pause time) and 10% when it is moderate (400 seconds of pause time). The adjustment value does not affect the results. Figure 6 presents other performance measures with the same simulation environment as in Figure 5. Because the data transmission from the CBR sources is based on UDP, some data packets can be lost and Figure 6(a) shows the ratio of received data. Acceptance ratio of *ROUTE_REQ* messages is presented in Figure 6(b). A *ROUTE_REQ* is dropped when an intermediate node has lower battery level than its threshold value ($E_r < Th_r$). With smaller adjustment value (d=0.1), a node will drop *ROUTE_REQ* messages more frequently and the corresponding overhead is increased. It is noted that the ratio is not 100% even with the DSR algorithm. Since each *ROUTE_REQ* message has a limited *TTL (time to live)*, a node may drop *ROUTE_REQ* message when the TTL value expires.



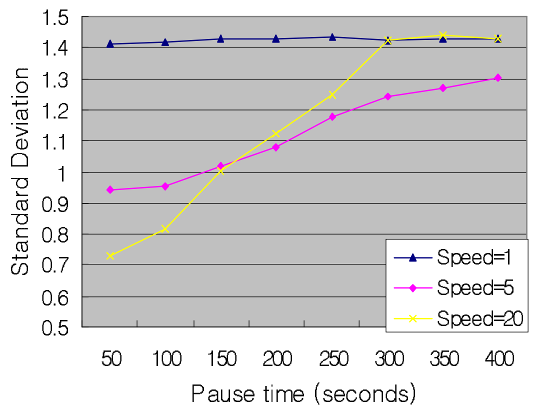
(a) Standard deviation



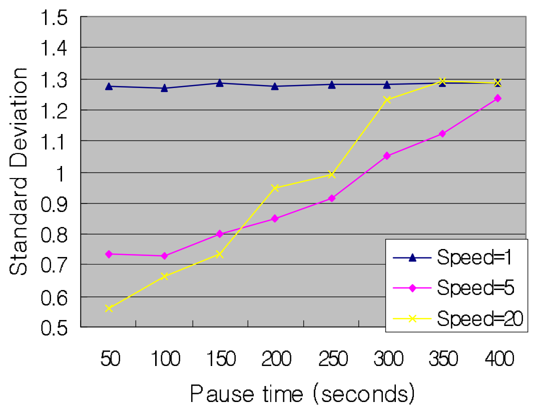
(b) Peak -to-Mean ratio

Figure 7. Distribution of Energy Consumptions with Faster Node Speed
(*MOBILITY-WP-MIN-SPEED* ~ *MOBILITY-WP-MAX-SPEED* = 0~20 meters/second)

When the maximum node speed is faster (*MOBILITY-WP-MAX-SPEED* or 20 meters/second), LEAR still outperforms DSR, but the benefit is reduced to as much as 20% as shown in Figure 7. Effect of node speed is explored in more detail as shown in Figure 8. Figures 8(a) and (b) compare the energy distribution with different node speeds (*MOBILITY-WP-MAX-SPEED* is 1, 5 or 20 meters/second) with DSR and LEAR (d = 0.1), respectively. When the maximum node speed is 1 meter/second, the energy distribution is not affected by the pause time. However, with higher node speed, the energy balance is affected by the pause time. With smaller pause time, higher node speed results in better distribution for both LEAR and DSR. However, with larger pause time, higher node speed makes the mobile nodes more stationary and thus results in worse distribution. For example, with pause time of 300 seconds, a node stays at its initial position for 300 seconds, moves to a random position at a faster speed (maximum 20 meters/second) and stays there rest of the simulation time. Thus a node statistically tends to be more stationary than that with the case of slower node speed (maximum 5 meters/second), resulting in worse energy distribution.



(a) DSR Algorithm



(b) LEAR Algorithm (d=0.1)

Figure 8. Distribution of Energy Consumptions with Different Node Speeds

5 Related Work

General idea of the energy-efficient operation in mobile devices is to power down individual components when they are idle, for example, reducing the CPU clock speed [23], spinning down the internal disk, turning off screen lighting, or making the radio subsystem to sleep. For example, *Palm Pilot* from *Palm, Inc.* consumes 80-90mA when CPU is busy, serial port is active and backlight is on, while it consumes as little as 0.13-0.3mA when the device is sleeping [24]. *Lucent*'s *WaveLAN-II* [20] implementing *IEEE 802.11 wireless LAN standard* consumes 250mA and 300mA when receiving and transmitting, respectively, while consumes only 9mA when it is in sleeping mode. This power reduction is achieved by supporting power down modes in the wireless communication hardware. For example, a *WaveLAN* radio device implementing *IEEE 802.11* [21] has two power down states (*Awake* and *Doze* states) and the corresponding protocols put the radio device into a low-power state when the device is not active. *Active mode* of operation requires the device in Awake state whereas a device in *power save mode* will switch between the Awake and Doze states. When in Doze state, the device will not monitor the medium. It is not able to receive a data packet but wakes up at regular intervals to check whether there is any packet destined to it. Table 6 summarizes those power down modes used in IEEE 802.11 and Bluetooth[5] [22] wireless LAN protocols as well as typical power consumption values of the devices implementing the protocols. *PAMAS (Power-Aware Multiple Access Protocol with Signaling)* is another MAC layer protocol for ad hoc networks where a node powers its radio off if it is not actively transmitting or receiving packets [25].

Table 6. Power Down States and Modes[6]

IEEE 802.11			Bluetooth	
Hardware State	Mode of Operation		Mode of Operation	Hardware State
Awake	Active	Transmit (300mA)	Active (40-60mA)	Connection
		Receive (250mA)		
		Idle or Listen (230mA)		
	Power Save			
			Sniff	
Doze			Hold	
	Sleep (9mA)		Park	
			Standby (0.55mA)	Standby

There also has been active research on energy awareness implemented in transport or application layer. By abstracting power management to a higher level, application-specific information can be exploited to reduce the amount of idle time by judiciously suspending it. *Stemm et al.* measured the energy consumption by wireless network interfaces in hand held devices and showed via simulation that it can be greatly reduced by incorporating application-specific information with mail and web access applications [26]. *Kravets and Krishnan* proposed a transport level functionality for managing the suspend/resume cycle of the node's network interface and allowed an application to control the communication device using the functions [2]. *BECA (basic energy-conserving algorithm)* proposed in [27] uses application-level information to turn off the radio more frequently and for a longer duration. Those studies found that the wireless network subsystem is in idle state most of the time and performs listen operations a lot compared to receive and transmission. Even though receive and transmission requires larger energy than listening, total energy consumption is dominated by listening due to the longer time in idle state. However, their analysis is based on static environment or single-hop wireless environment with fixed infrastructure. In case of multi-hop mobile ad hoc networks, a local application should not suspend the radio device because it may have to provide the connectivity for others' behalf. If there are many neighboring nodes within its radio range, the idle time is reduced and the receive will dominate the time span.

[5] Bluetooth supports lower bit rate of 768 Kbps and shorter radio range up to 10 meters or 100 meters depending on transmitter's power compared to IEEE 802.11-compliant WaveLAN-II's 2 Mbps bit rate and 250 meters radio range.
[6] Power consumption values included in the table are for the IEEE802.11-compliant WaveLAN-II from Lucent and Bluetooth wireless interface from Nokia, respectively [20,22].

6 Conclusions and Future Works

An energy-aware routing for ad hoc networks, called Local Energy-Aware Routing (LEAR), was introduced. LEAR achieves balanced energy consumption based only on local information, thus removes the blocking property of other energy-aware routing algorithms proposed elsewhere. Another important advantage of LEAR is its simplicity and it can be easily integrated into existing ad hoc routing algorithms without affecting other layers of communication protocols. Simulation results show that energy usage is better distributed with the proposed LEAR algorithm as much as 35% compared to the DSR algorithm. To the best of our knowledge, this is the first work to explore the balanced energy consumption in a realistic environment where routing algorithms, mobility and radio propagation models are all considered.

This paper does not include the simulation results with GEAR, is not realistic due to the indefinite delay at the decision-making nodes. However, since GEAR can provide ideal energy distribution, it is worthwhile to compare the results with those with LEAR. We leave this as our future work. We also want to refine and optimize the LEAR algorithm by combining with the concept of *APR (alternative path routing).* LEAR, in essence, selects an alternative path when a current path is not energy-rich. There have been numerous research works on APR [28, 29], where the main goal is to continually provide a path even when an old one is not valid due to node movement. Another future research topic is to explore the possibility of applying the proposed idea in this paper to the broadcast type of network traffic in mobile ad hoc networks. Tremendous data traffic may be generated as explained in [30] and thus the battery energy of some nodes can be easily depleted.

References

[1] G. Forman and J. Zahorjan, "The challenges of mobile computing," *IEEE Computer*, Vol. 27, No. 4, pp. 38-47, Apr. 1994.

[2] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication," *International Conference on Mobile Computing and Networking (MobiCom'98)*, Oct. 1998.

[3] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, edited by T. Imielinski and H. F. Korth, Kluwer Academic Publishers, 1996.

[4] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM*, 1994.

[5] R. Castaneda and S. R. Das, "Query Localization Techniques for On-demand Routing Protocols in Ad Hoc Networks," *International Conference on Mobile Computing and Networking (MobiCom'99)*, pp. 186-194, 1999.

[6] J. Broch, D. A. Maltz and D. B. Johnson, "Supporting Hierarchy and Heterogeneous Interfaces in Multi-Hop Wireless Ad Hoc Networks," *Workshop on Mobile Computing (I-SPAN)*, Jun. 1999.

[7] C.-K. Toh, "Long-lived Ad-Hoc Routing based on the Concept of Associativity," *Internet-Draft of IETF MANET Working Group*, Mar. 1999.

[8] C. E. Perkins, E. M. Royer and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," *Internet-Draft of IETF MANET Working Group*, Jul. 2000.

[9] W.-H. Liao, Y.-C. Tseng, and J.-P. Sheu, "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks", *Telecommunication Systems*, (to appear).

[10] J.-C. Cano and Pietro Manzoni, "A Performance Comparison of Energy Consumption for Mobile Ad Hoc Network Routing Protocols," *Eighth International Symposium on Modeling, Analysis and Simulation of Computer andTelecommunication Systems (MASCOTS 2000)*, Aug. 2000.

[11] S. Singh, M. Woo, and C. S. Raghavendra, “Power-Aware Routing in Mobile Ad Hoc Networks,” *International Conference on Mobile Computing and Networking (MobiCom'98)*, pp.181-190, Oct. 1998.

[12] J.-H. Chang and L. Tassiulas, “Energy Conserving Routing in Wireless Ad-hoc Networks,” *The Conference on Computer Communications (IEEE Infocom 2000)*, pp.22-31, 2000.

[13] R. Ramanathan and R. Rosales-Hain, “Topology Control of Multihop Wireless Networks using Transmit Power Adjustment,” *The Conference on Computer Communications (IEEE Infocom 2000)*, pp.404-413, 2000.

[14] I. Stojmenovic and X. Lin, “Power-aware localized routing in wireless networks,” *International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 371-376, 2000.

[15] X. Zeng, R. Bagrodia and M. Gerla, “GloMoSim: A Library for the Parallel Simulation of Large Scale Wireless Networks,” *Parallel and Distributed Simulation Conference (PADS)*, 1998.

[16] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu and J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols,” *International Conference on Mobile Computing and Networking (MobiCom'98)*, Oct. 1998.

[17] S. Corson and J. Macker, “Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations,” *RFC 2501 of IETF NetworkWorking Group* , Jan. 1999.

[18] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek and M. Degermark, “Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks,” *International Conference on Mobile Computing and Networking (MobiCom'99)*, pp. 195-206, 1999.

[19] R. Bagrodia, et al, “PARSEC: A Parallel Simulation Environment for Complex Systems,” *IEEE Computer*, Oct. 1998.

[20] A. Kamerman and L. Monteban, “WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band,” *Bell Labs Technical Journal*, pp. 118-133, Summer 1997.

[21] Hagen Woesner, Jean-Pierre Ebert, Morten Schlager, and Adam Wolisz, “Power-Saving Mechanisms in Emerging Standards for Wireless LANs: The MAC Level Perspective,” *IEEE Personal Communications*, Vol. 5, Issue 3, pp. 40-48, Jun. 1998.

[22] “Complete Bluetooth Tutorial,” http://infotooth.tripod.com/tutorial/complete.htm.

[23] M. Weiser, A. Demers, B. Welch and S. Shenker, “Scheduling for Reduced CPU Energy,” *Operating System Design and Implementation (OSDI) Conference*, 1994.

[24] R. Nicholson, “Short and Unofficial FAQ on PalmOS Handhelds,” http://www.nicholson.com/rhn/palm.html

[25] S. Singh and C. S. Raghavendra, “Pamas – power aware multi-access protocol with signaling for ad hoc networks,” *ACM Computer Communication Review*, Jul. 1998.

[26] M. Stemm and R. H. Katz, “Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices,” *International Workshop on Mobile Multimedia Communications (MOMUC-3)*, Sep. 1996.

[27] Y. Xu, J. Heidemann, and D. Estrin, “Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks,” *Research Report*, No. 527, USC/Information Sciences Institute, Oct., 2000.

[28] A. Nasipuri and S. R. Das, “On-Demand Multipath Routing for Mobile Ad Hoc Networks,” *International Conference on Computer Communication and Network (ICCCN'99)*, Oct. 1999.

[29] M. R. Pearlman, Z. J. Hass, P. Sholander and S. S. Tabrizi, “On the Impact of Alternate Path Routing for Load Balancing in Mobile Ad Hoc networks,” *The First Annual Workshop on Mobile Ad Hoc Networking & Computing (MobiHOC 2000)*, Aug. 2000.

[30] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad hoc Network", *Int'l Conf. on Mobile Computing and Networking (MobiCom'99)*, pp. 151-162, 1999.