An On-demand Minimum Energy Routing Protocol for a Wireless Ad Hoc Network

Sheetalkumar Doshi[a] **Shweta Bhandare**[b] **Timothy X Brown**[a,b]
doshi@colorado.edu *bhandare@colorado.edu* *timxb@colorado.edu*
[a]Department of Electrical Engineering, University of Colorado, Boulder, CO, USA
[b]Department of Interdisciplinary Telecommunications, University of Colorado , Boulder, CO, USA

A minimum energy routing protocol reduces the energy consumption of the nodes in a wireless ad hoc network by routing packets on routes that consume the minimum amount of energy to get the packets to their destination. This paper identifies the necessary features of an on-demand *minimum energy routing protocol and suggests mechanisms for their implementation. We highlight the importance of efficient caching techniques to store the minimum energy route information and propose the use of an 'energy aware' link cache for storing this information. We compare the performance of an* on-demand *minimum energy routing protocol in terms of energy savings with an existing* on-demand *ad hoc routing protocol via simulation. We discuss the implementation of Dynamic Source Routing (DSR) protocol using the Click modular router on a real life test-bed consisting of laptops and wireless Ethernet cards. Finally we describe the modifications we have made to the DSR router to make it energy aware.*

I. Introduction

Wireless ad hoc networks usually consist of mobile battery operated computing devices that communicate over the wireless medium. These devices need to be energy conserving so that the battery life is maximized. While research continues to reduce the energy consumption for the CPU, user interface and storage for the devices, the energy for transmission of a packet in the wireless channel remains quite significant and may turn out to be the highest energy-consuming component of the device. Traditional communication theory focuses on physical and link layer mechanisms for reducing energy consumption. Here we explore minimum energy routing protocols. For such a protocol design, we need to look away from the traditional minimum hop routing schemes and design new routing schemes that take the transmission energy into consideration for choosing the appropriate route. Efficient minimum energy routing schemes can greatly reduce energy consumption and lead to a longer battery life of the device.

Ad hoc routing protocols can be broadly classified as table driven routing protocols and source initiated on-demand routing protocols [21]. The first approach uses a routing table which is maintained via periodic updates from all the other nodes in the network irrespective of the fact that the network may not be active in terms of data traffic. The on-demand approach, on the other hand, sends out requests for routes to the destination only if the source node has data-packets which are to be sent to the destination. Brown et al. [3] show that the table driven schemes are more expensive in terms of energy consumption as compared to the on-demand schemes because of the large routing overhead incurred in the former. Therefore we focus on on-demand routing protocols in this study.

Prior studies have been aimed at designing minimum energy routing protocols. Subbarao [24] suggests a minimum power routing scheme that has been designed using the table-driven approach. Singh et al. [23] introduce power aware cost metrics for routes and design routing schemes that minimize these metrics. They also suggest MAC layer modifications which power down the nodes when inactive to obtain energy savings. The scheme suggested by Ramanathan et al. [19] brings about power savings by using transmit power adjustment to control the topology of a multi-hop wireless network. Rodoplu et al. [20] develop a distributed position based network protocol that uses location information to compute a multi-hop route which minimizes the energy to deliver the packet. Chang et al. [5] propose algorithms to select routes and corresponding power levels in a static wireless ad hoc network such that the system lifetime (in terms of battery life) is maximized. Xu et al. [27] suggest a location information based energy conserving routing algorithm that works above the ad hoc routing agent protocol to bring about energy conservation in the network by powering down

intermediate nodes while still maintaining connectivity. Li et al. [12] develop an online approximate power aware routing algorithm to maximize the lifetime of the ad hoc network, which involves choosing between the minimal power consumption path and the path that maximizes the minimum residual power in the network. Chen et al. [6] suggest a distributed coordination technique for a multi-hop ad hoc wireless network that reduces energy consumption without significantly reducing the capacity and connectivity of the network. Energy savings are achieved by keeping certain chosen coordinator nodes active while the other nodes in the network are in a power save mode. Shih et al. [22] introduce the design of physical layer aware protocols, algorithms and applications that minimize energy consumption of the system and individual nodes. The effect of the entire hardware architecture on the design of protocols and algorithms is considered. Brown et al. [4] study the fairness of different power aware routing objectives. Although simply minimizing the power of each route does not maximize any particular network energy performance metric, it generally reduces network energy consumption and for reasons of simplicity we focus on it here. In this paper we enumerate necessary features of a minimum energy routing protocol. These features will guide a minimum energy routing protocol design.

In Section II we present an overview of the DSR protocol and the 802.11 MAC layer protocol. We describe the energy model used for our study in Section III. Section IV discusses the factors that can be used to reduce energy consumed by the existing ad hoc routing protocols. We discuss the required features of a minimum energy routing protocol in Section V. In Section VI, we suggest mechanisms that can be used to derive a 'minimum energy routing' version of existing ad hoc routing protocols such as the Ad-Hoc On-demand Distance Vector (AODV) protocol and the Dynamic Source Routing (DSR) protocol. We test a minimum energy routing version of DSR on the network simulator (ns) framework and compare its performance with the existing version of DSR in Section VII. We discuss the choices for the hardware test-bed in Section IX and describe the implementation of the DSR protocol using the Click modular router in Section X. Finally in Section XI, we describe the changes that we have done to make this protocol energy aware. Although this paper focuses on DSR and 802.11b, the principles apply to other routing and MAC protocols.

II. Overview of the existing protocols

We begin by outlining the existing version of DSR [14] and the 802.11 Medium Access Control protocol [13] since these will provide a reference for discussing the necessary minimum energy routing features and serve as a baseline for our performance comparisons. Other routing protocols that could have been considered are the Destination Sequenced Distance Vector routing protocol (DSDV) [17] and the Ad-hoc On-Demand Distance Vector routing protocol (AODV) [18]. However, the DSDV protocol is a table driven routing protocol and among the DSR and the AODV protocols, the DSR protocol has been implemented on a wireless ad hoc network consisting of laptops with IEEE 802.11 WaveLAN cards [16]. This prompted us to choose the DSR protocol as a baseline to integrate our energy aware features in a working ad hoc network.

II.A. The Dynamic Source Routing protocol mechanism

The Dynamic Source Routing protocol is an on-demand routing protocol that is based on the concept of source routing. The protocol maintains a route cache in each node which is updated as new routes are learned. When a node has a packet to send to some destination, it looks in the cache to determine if it already has a route to the destination. If there are multiple routes to the destination in the cache, the routing logic selects the minimum hop route. The node then inserts this route in the routing header of the data-packet and sends it to the MAC layer for transmission over the medium. In case the source node has no route to the destination in its cache, it initiates a route discovery process: it broadcasts a *Route Request* packet containing the destination node address, the source node address and a unique *Request ID*. Each node that receives the route request checks its cache for a route to the destination in case the 'reply from cache' option is enabled in the routing agent logic. If this option is disabled or if no route is found in the cache, the node adds its own address to the address chain in the route request packet and rebroadcasts the packet. However, in case the node has already heard a route request with the same request ID earlier, the node does not rebroadcast the packet. This continues till the packet reaches the destination node or, if the 'reply from cache' option is enabled, till it reaches a node that knows a route to the destination. Here again, in case there are multiple routes, the minimum hop route is selected. This

node makes up a *Route Reply* packet and inserts the full source route to the destination in the route reply packet and uses the reverse route from itself to the source node to route this reply packet to the source node.

Route maintenance is carried out in case the links in the routes being used break due to channel fluctuations and node mobility causing the received power to fall below the receiver sensitivity threshold. *Route Error* packets are generated at a node in case the link layer reports a broken link during a data-packet transmission. This route error packet is routed to the source node of the data-packet through the intermediate nodes in the route so that they can update the caches by removing the hop in error and truncating all other routes that contain that hop till that point. Further details about the DSR protocol can be found in [14].

II.B. The 802.11 MAC layer protocol

The basic access method in the 802.11 MAC protocol is the Distributed Coordination Function (DCF). The DCF defines two ways of transmitting data frames over the medium: the so called *two frame exchange* and the *four frame exchange* mechanism.

In the *two frame exchange*, the station transmits a DATA frame if the medium is determined to be idle for an interval that exceeds the Distributed Inter Frame Space (DIFS). If the medium is busy, the station waits till the channel is idle for a DIFS and then generates a random back-off period for an additional deferral time before trying to transmit again. The receiver sends immediate positive ACK frames to the sender after the successful reception of each data frame.

The *four frame exchange* scheme transmits special short Request To Send (RTS) and Clear To Send (CTS) frames prior to the transmission of the actual DATA frame. The transmitter sends an RTS frame after the channel has been idle for a time interval exceeding DIFS. On receiving an RTS frame the receiver responds with a CTS frame. In case the CTS frame is not received within a predetermined time interval, the sender retransmits the RTS. After the successful exchange of the RTS and CTS frames the DATA frame is sent by the sender and the ACK is transmitted by the receiver after the successful reception of the DATA frame. Using this four frame exchange mechanism leads to a reduced probability of collisions. However this reduced probability of collisions is achieved at the expense of an increased energy overhead involved with the exchange of RTS and CTS frames, which can be significant for short data frames.

In both schemes, all frames are transmitted at full power. The network designer can decide which scheme is to be used for the DATA frame transmission by setting a programmable packet size threshold. Packets smaller than this threshold are transmitted using the two frame exchange, otherwise the four frame exchange is used. Further details regarding the 802.11 protocol can be found in [13]. The energy consumption characteristics of the 802.11b cards that implement the 802.11 protocol are carefully observed in [8]. The 802.11b cards are not the best choice in terms of achieving energy efficiency, but we use them for our protocol development due to their widespread support and availability.

III. Energy model used for this study

In this section we present the energy model used for our analysis. This model helps us justify the required features of a minimum energy routing protocol.

The energy expended in sending a data-packet of size D bytes over a given link can be modeled as

$$E(D) = K_1 D + K_2 \tag{1}$$

Feeney et al. [8] propose a similar model in their study to describe the *per packet* energy consumption. In our model,

$$K_1 = (P_t^{packet} + P^{back})/BR \tag{2}$$

$$K_2 = ((P_t^{MAC} D^{MAC} + P_t^{packet} D^{header})/BR) + E^{back} \tag{3}$$

where P^{back} and E^{back} are the background power and energy used in sending the data-packet, P_t^{MAC} is the power at which the MAC packets are transmitted, D^{MAC} is the size of the MAC packets in bytes, D^{header} is the size of the data-packet trailer and header, P_t^{packet} is the power at which the data-packet is transmitted and BR is the transmission rate in Bytes/sec. In order to simplify our analysis, we assume P^{back} and E^{back} to be zero in our study.

IV. Why we think we can do better

A number of factors can be exploited to reduce the energy consumed by the existing ad-hoc routing protocols. In this section, we explain each of the factors.

IV.A. Required energy decreases rapidly with distance.

For a given threshold power P_r, the minimum transmit power P_t required for successful reception, assuming no fading, can be given as

$$P_t(d) = P_r d^n / K \tag{4}$$

where d is the distance between the two nodes, n is the path loss exponent and K is a constant. Typically n takes the value of 4. In case of maximum power minimum hop routing used in typical ad hoc routing protocols like the current version of DSR, this transmit power is fixed to 280mW (as per wireless LAN 802.11 specifications). Using (1), (2) and (3), the minimum

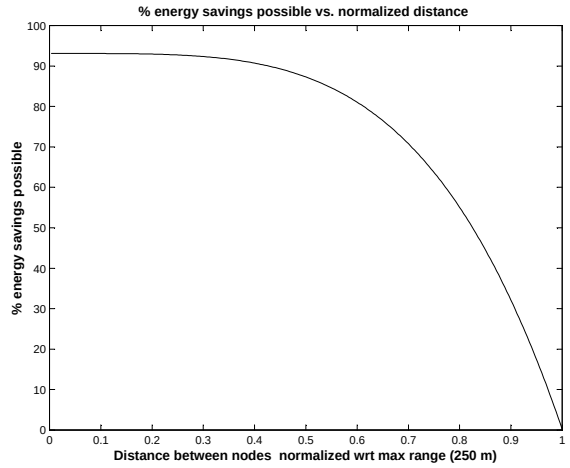


Figure 1: Energy savings possible using transmit power control

transmission energy required for successful reception in terms of P_t and data-packet size D can be given as

$$E_t(D, P_t) = K_1' P_t(D + D^{header}) + K_2 \quad (5)$$

and substituting the value of P_t obtained from (4) in (5), we get

$$E_t(D, d) = K_1''(D + D^{header})d^4 + K_2 \quad (6)$$

Typical values for K_1', K_1'' and K_2 in a two frame exchange 802.11 MAC environment with ACKs sent at full power and a 2Mbps bit rate are $4\mu s/byte$, $2.8\times10^{-10}\mu J/(byte\ m^4)$ and $42\mu J$ respectively. The transmission energy for the current version of the protocols (E_{max}) is fixed for a fixed data-packet of size D bytes and can be given as

$$E_{max}(D) = K_3 D + K_2 \quad (7)$$

where K_3 has the value of $1.162\mu J/byte$.

The energy savings that can be obtained by using the minimum transmit power instead of the fixed maximum power for the data-packet transmission is given as

$$S(D, d) = E_{max}(D) - E_t(D, d) \quad (8)$$

The plot of these savings vs. the distance between the two nodes for a data-packet size of 512 bytes and a two frame exchange MAC layer scheme can be seen in Fig. 1. This gives us a fair idea about the achievable energy savings using power control for data-packet transmission.

IV.B. Using multi-hop routes saves energy.

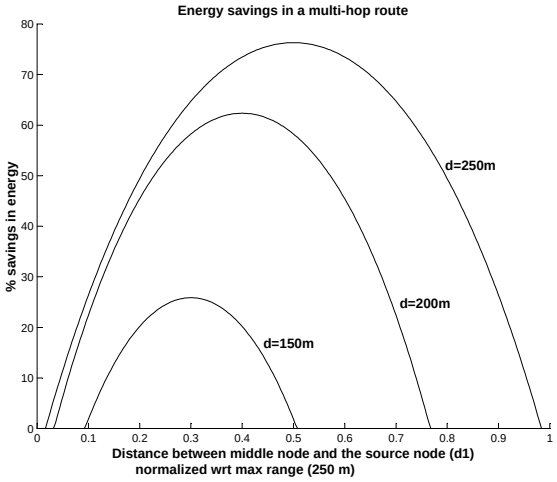


Figure 2: Effect of fixed energy overhead per hop on energy savings obtained using multi-hop routes

Consider a case where the minimum hop routing protocol employs transmit power control. In this case there are 3 nodes a, b, c in a straight line and the minimum hop routing chooses to route data-packets directly from a to c. If the multi-hop route a-b-c is chosen to transmit the packets instead, i.e. b is used as a relay node, the total transmit energy required would be

$$E_{multi}(D, d, d_1) = E_t(D, d_1) + E_t(D, d - d_1) \quad (9)$$

where d_1 is the distance from Node a to Node b and d is the distance from Node a to Node c.

The savings, $S(D, d, d_1)$, obtained by going the multi-hop route can be written as

$$S(D, d, d_1) = E_t(D, d) - E_{multi}(D, d, d_1) \quad (10)$$

i.e.

$$S(D, d, d_1) = E_t(D, d) - E_t(D, d_1) - E_t(D, d - d_1) \quad (11)$$

However the energy savings obtained by going multi-hop depends on the value of the fixed energy overhead K_2 and the distance from Node a to Node c. Fig. 2 shows a plot of % energy savings vs d_1 for different values of d fixing K_2 as $0.05E_{max}$ and shows that for larger d the savings is large and the savings decrease as d decreases. Further, if Node b is too close to Node a or Node c, the multi-hop route consumes more energy than the direct hop route.

V. Required Features of a minimum energy routing protocol

This section enumerates the required features of a minimum energy routing protocol. We also show that the existing versions of the on-demand routing protocols do not support most of these required features and hence justify our argument that existing on-demand protocols cannot qualify as minimum energy routing protocols.

V.A. Energy-based routing cost computation

Minimum energy routing protocols should take into consideration the energy cost of a route while choosing a route. The energy cost of a link in a route is the sum of the energy required for transmission of a data-packet over that link and the additional signalling and packet processing cost. The existing on-demand routing protocols employ a hop metric for choosing routes and not an energy metric.

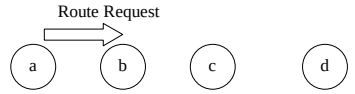
V.B. Transmit power control

For a given threshold power P_r, the minimum transmit power P_t required for successful reception varies with distance is given by (4). To gain maximum energy savings, the minimum energy routing protocol should transmit the data-packet at power P_t instead of the fixed transmit power. This can be achieved by employing dynamic transmit power control on the link. Now if dynamic transmit power control is employed, the energy cost of each link can be computed using (2) and (3). Thus knowing K_1, K_2, and the value of P_t for each link, the minimum energy routing protocol should compute the link energy cost using (1) and should choose the route with the lowest sum of link energy costs.

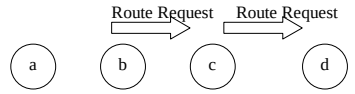
The existing on-demand protocols do not offer any mechanisms to compute and propagate the minimum transmit power value for each link of the route. Hence the dynamic transmit power control feature cannot be supported by the existing versions of the on-demand protocols.

V.C. Minimum energy route discovery

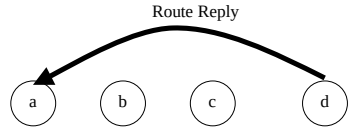
Consider a case as shown in Fig.3 where there are 4 nodes a, b, c, d in a straight line. Assume the minimum energy route from Node a to Node d is the multi-hop route a-b-c-d. Rodoplu at al. [20] show that the minimum energy routes in a network translate to multi-hop routes and the minimum energy routing protocol should be able to discover these minimum energy routes. The route discovery mechanisms of existing on-demand protocols are similar in the way the route discovery is initiated. For finding a route from Node a to Node d, the mechanisms initiate a Route Request packet broadcast from Node a. Assuming that this packet is heard by Nodes b and c, both nodes rebroadcast the packet. The packets broadcasted by Nodes b and c are heard by all nodes. However since Node c has already broadcasted the same request earlier, it ignores the request packet from Node b and Node d replies back to the requests it hears from Nodes b and c. Hence the on-demand routes discovered by Node a are a-b-d and a-c-d: The minimum energy route a-b-c-d is not discovered by the route discovery mechanism of the existing on-demand routing protocols.



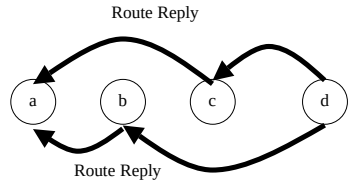
a) Node a sends a route request and this route request packet is heard by b, c and d.



b) Nodes b and c propagate this route request they hear from node a. However, node c does not propagate the route request that b broadcasts.



c) Node d sends a route reply a-d to node a.



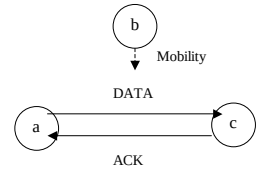
d) Node d also sends the replies a-c-d and a-b-d to node a.

Figure 3: Route Discovery in the existing version of the DSR protocol

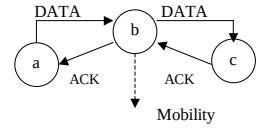
V.D. Tracking energy costs in the minimum energy routes.

In the current version of on-demand ad hoc routing protocols, route maintenance is carried out by the route error packets only when the links are broken. No route maintenance is done to indicate the change in the quality of a link. No mechanism updates the information about the changing power requirements that oc-

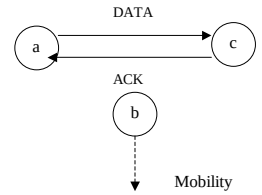
cur on that route due to node mobility. Even after the minimum energy routes are discovered, the changes in the energy costs of the links have to be tracked so that the energy expended is as close to the minimum value as possible. In case the energy cost of a certain link rises due to the nodes moving apart, the route may no longer be the minimum energy route. Therefore, these changes in the energy cost need to be conveyed to the source node, so that it can choose other lower energy routes as needed. Hence a minimum energy routing protocol must have a mechanism for tracking the energy cost changes in the routes. Mobility also causes the creation of new lower energy routes. For example as seen in Fig.4, with 3 nodes a, b, and c, the minimum energy route from Node a to Node c at one instance of time may be a-c. Now because of node mobility, Node b moves to a position between Node a and Node c making the new optimal minimum energy route from Node a to Node c as a-b-c. Therefore, to maintain minimum energy routing, additional route maintenance which goes beyond achieving basic connectivity is required. Hence this feature too is not supported by existing versions of on-demand routing protocols.



a) The link a-c is the minimum energy route. Node b is moving between Nodes a and c.



b) The minimum energy route is a-b-c. Node b routes packets through itself.



c) Node b moves away. The minimum energy route is now a-c.

Figure 4: Tracking and discovering minimum energy routes

V.E. Scalability

The overhead incurred by the minimum energy protocols should scale well with the number of nodes in the network. Minimum energy routes essentially translate to multi-hop routes and as the number of nodes in the network increase, the number of hops in the minimum energy route increase and so does the overhead in discovering and maintaining these routes.

VI. Mechanisms to implement these features

Existing versions of on-demand ad hoc routing protocols do not possess most of the required features of a minimum energy routing protocol as demonstrated in the previous section. This section describes mechanisms for the easy implementation of these features in the routing logic of the existing on-demand protocols.

VI.A. Link energy cost computation

Given the value of P_t in (5), the value of K_1 and K_2 in (1) can be computed. This power value P_{TXmin} is included along with the node ID in the route request packet information and the packet is re-broadcasted by the receiver node. The destination node reverses the route in the route request packet and inserts this power information for each hop in the routing header of the route reply packet to route the route reply packet to the source node using transmit power control at each hop. Each node in the reply path stores the value of this minimum transmit power required to get to the next hop, P_{TXmin}. The source node gets the exact power values for each hop from the route reply from which it can calculate the total energy cost of the route by using (5) and a suitable value for K_2.

VI.B. Transmit power control

This mechanism can be implemented by modifying the route request packet header to include the power at which the packet was transmitted by the source node. Let P_{TX} be this transmit power in dBm. The receiving node receives this packet at power P_{RX} in dBm. Let P_{Thresh} be the minimum power level required for a successful reception in dBW. The minimum power required for the transmission of the packet so that it is successfully received by the receiver (P_{TXmin}) can be then calculated in dBW by the receiving node as

$$P_{TXmin} = P_{TX} + P_{Thresh} - P_{RX} + M \qquad (12)$$

where M is a margin to overcome the problem of unstable links due to channel fluctuations and node mobility. The receiver node can read the value of P_{TX}

from the header of the received packet. For routing the data-packet, in case of protocols like AODV [18], nodes of the network can simply look up the value of the minimum transmit power from their routing table and transmit the data-packet at the controlled power. For source routing on-demand protocols like DSR [14], this minimum transmit power value for each hop is included in the routing header of the data-packet by the source node and each node forwarding the packet simply looks up the next hop in the source route and the minimum power required to get there and transmits the packet at the controlled power level.

VI.C. Route discovery of minimum energy routes

This mechanism enables the node to store the route power information it hears in the route request packets. The node then snoops on route replies not directed to the node and checks if it lies on a lower energy path than advertised in the route reply using the stored route power information and the route reply power information. In case it lies on a lower energy path, the node sends a gratuitous route reply with the lower energy path to the source. In this manner the nodes can discover the minimum energy route.

VI.D. Minimum energy route maintenance and link energy cost tracking

This mechanism enables the nodes in a data-packet's source route to sense the power changes in the source route as the data-packet is forwarded to the destination. Each node senses the power at which it receives the data-packet and computes the new minimum transmit power $P_{TXminnew}$ and the new minimum transmit energy $E_{TXminnew}$ required by the previous node to successfully transmit the packet to it using (12), (5) and (1) respectively. The current node compares this energy value to the original value E_{TXmin} calculated from P_{TXmin} (which is advertised in the source route in case of DSR and is present in the routing table in case of AODV) and if

$$|E_{TXminnew} - E_{TXmin}| > T \qquad (13)$$

where T is a threshold value in dB, the node sets a flag in the routing header of the data-packet about the changed energy cost of the link and writes this new power value in the routing header. Now once the destination gets the data-packet, it checks the flag in the routing header of the data-packet and sends a gratuitous route reply packet containing the route with the new power information to the source. The intermediate nodes and the source node update their cache/routing table with this new power information and thus keep track of the energy cost changes of the links of the route.

In case of node mobility, lower energy routes can form after the initial route discovery of a low energy route on which the data-packet flow is being sent. The route maintenance logic should discover and take advantage of these routes. This mechanism modifies the MAC ACK header to include the transmit power information of the ACK frame. A node can snoop on a data-packet not directed to itself and then on the ACK for the data-packet from the receiver. The node computes the energy cost from itself to the transmitter node (E_{toTX}) from the data-packet power information and the energy cost from itself to the receiver node (E_{toRX}) from the ACK power information. Let E_{TXmin} be the advertised energy cost to get from the transmitter to the receiver node. Now if

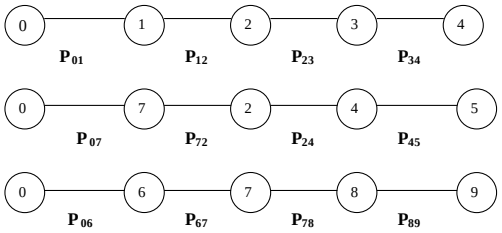
$$E_{toRX} + E_{toTX} < E_{TXmin} \qquad (14)$$

the node understands that it lies on a lower energy path between the transmitter and the receiver nodes. It then sends out a gratuitous route reply to the source of the data-packet informing it about the lower energy route (DSR) or to the transmitting node (AODV). This mechanism enables the nodes to keep track of the optimum minimum energy route over time.

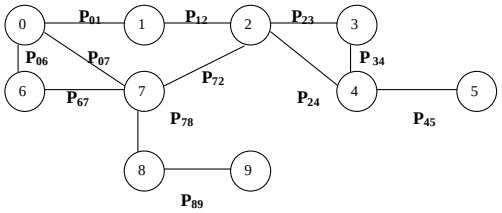
VI.E. Ensuring scaling of the minimum energy routing protocol

To ensure scaling, the protocol must converge to the minimum energy route keeping the overhead bounded. This can only be done if the routing and power information obtained in the route replies is stored and processed in an intelligent manner so that every route reply leads to a clearer topology view of the network at each node. This calls for storing the routing and the power information in a unified graph data structure of the node's current view of the network topology. Hu et al. [10] use a similar structure, the so-called *link cache* for storing the information obtained from the route replies. We suggest the use of a similar data structure, the *energy aware link cache* as shown in Fig. 5, where the cost of each link is set as the energy cost. The effectiveness of the link cache can be observed from the fact that the path cache will return only one path from Node 0 to Node 3, whereas the link cache will examine the six possible paths and choose the most efficient of them. The energy aware link cache saves the information about the energy cost of each link (K_1 and K_2 from (1)) from the route reply in an array and the well-known Dijkstra's shortest

path algorithm is used to find the current minimum energy cost path through the graph to the destination node. Thus each route reply contributes in building up a better topology view of the network and thus obtaining a better energy cost graph.



(a) Path Cache with power information



(b) Energy Aware Link Cache

Figure 5: Different types of caches. P_{ij} is the minimum transmit power required to get from Node i to Node j

VI.F. Bi-directional links and minimum transmit power computation

The previous section assumes that the links between the nodes are bi-directional: the P_{thresh} and the maximum transmit power is the same in both directions. This assumption may not always be true i.e. P_{thresh} and maximum transmit power may not be the same for all the nodes. Hence, for a link between nodes a and b, P_{TXmin} for the link a-b may not be equal to P_{TXmin} for the link b-a. In case a node c wishes to estimate the P_{TXmin} value using (12) for c-b when it snoops on packets transmitted by b to a, it cannot do so without knowing the P_{thresh} value of node b. To overcome this, the P_{thresh} and maximum transmit power values of the nodes need to be advertised. Thus node c now can use node b's P_{thresh} and maximum transmit power value to calculate P_{TXmin} for the link c-b using (12).In our implementation, we assume that P_{thresh} and the maximum transmit power value are the same for all nodes.

VI.G. Costs of the modifications to the existing protocol logic

Modifying the existing protocol logic in order to implement the minimum energy routing features will incur some cost. This section enumerates the costs of each feature in terms of the types of modifications required.

Table 1: Classification of the costs of implementing minimum energy routing protocol features

Required Features of a minimum energy routing protocol	COSTS OF MODIFICATIONS		
	Routing software modification cost	802.11 Firmware modification cost	Radio hardware modification cost
A. Link energy cost computation	Yes	No	No
B. Transmit power control	Yes	Yes	Yes
C. Route Discovery of minimum energy routes	Yes	No	No
D. Minimum energy route maintenance and energy cost tracking	Yes	Yes	No
E. Link Cache	Yes	No	No

The cost of modifications can be classified as follows:

- *Routing software modification cost*: This reflects the cost of the changes that have to be made in the routing software of the existing protocol in order to support the feature.
- *802.11 Firmware modification cost*: This reflects the cost of the changes that have to be made in the 802.11 Firmware of the existing protocol in order to support the feature.
- *Radio hardware modification cost*: This reflects the cost of the changes that have to be made in the wireless Ethernet card hardware in order to support the feature.

Table 1 shows the classification of the costs for each of the features. The *link energy cost computation* feature needs changes only in the routing header to include the transmit power information of the packet so that the energy cost of the link can be calculated at the receiver node. Hence only the routing software needs to be modified to implement this feature. The *transmit power control* feature requires dynamically setting the transmit power of the wireless Ethernet card for each

packet to be sent. This will require changes in the radio hardware of the card as well as routing software and 802.11 Firmware changes. The *minimum energy route discovery* feature will require only routing software changes as it only involves storage and retrieval of the energy cost information of the links in the network. The *minimum energy route maintenance and energy cost tracking* feature requires transmit power information to be included in the MAC ACK packets which needs 802.11 Firmware modifications along with routing software changes. Finally, the *scalability* feature will need changes to be done only in the routing software to implement the link cache.

VII. A minimum energy routing version of DSR on the ns simulator

We implemented the mechanisms discussed in the previous section in the DSR protocol logic in the network simulator (ns) to obtain a *minimum energy routing* version of the DSR protocol. We used the network simulator (ns) version 2.1b6 [7] for our simulations. We used the DSR version with the promiscuous tap disabled. The energy model we used was the one bundled with the ns-2.1b6 package with the receiving cost of a packet set as zero in order to simplify our analysis. We employed the two frame exchange scheme for the MAC layer data transmission by setting the dot11RTSThreshold parameter to a value greater than the simulated data-packet size.

We introduced a few other changes in the DSR code other than the mechanisms discussed above to obtain the minimum energy routing protocol version. They were:

- Cache replies off: Maltz et al. [15] discuss the effect of on-demand behavior in ad hoc routing protocols and state in their results that in a typical simulation scenario, the majority of the route reply packets are based on cached data and only 59% of those replies carry valid routes. Hence for minimum energy routing where freshness of routes is even more important, we disabled route replies from the cache.

- MAC layer ACK power control: In order to reduce the signaling cost per hop, we employed transmit power control of the MAC ACK packets on the link.

For a clear analysis of the energy expended by the nodes during the simulation, we categorized the energy consumed into various components to understand the effect of the different routing modifications/options on these components. Hence the total energy consumed (E_{total}) was broken up as follows:

- Energy expended in DSR routing packets ($E_{routing}$)

- Energy expended in MAC signalling packets (E_{mac})

The data-packet energy was divided further into the following parts :

- Energy consumed by the MAC header (E_{machdr})

- Energy consumed by the routing header ($E_{routinghdr}$)

- Energy consumed by the data payload ($E_{datalen}$)

Here we introduce a concept called the God energy: let us assume that nodes are given the precise routing information so that they have a perfect picture of the network topology. Each node then can determine the path that takes up the least energy to get a packet to its destination and the data is sent without any overheads. This energy is termed as the God energy of that packet (i.e. the minimum cost in terms of energy to send the data packet without using signaling packets or packet headers). The routing efficiency of the protocol in terms of energy can be measured with respect to the God energy, E_{God}. $E_{datalen} - E_{God}$ can be considered as the routing inefficiency. These energies were logged into the trace file by modifying the trace code to include these fields.

The ns implementation of DSR comes with the path cache data structure. We implemented the link cache data structure as discusses in Section V.E. We experimented with both types of caches to observe which of these satisfy the criteria in Section V.E.

We compared the performance in terms of energy savings of three versions of the DSR protocol viz.

- DSR with link cache: This version of DSR is obtained by using a link cache data structure defined in [10] to store the route.

- DSR with link cache with transmit power control and power aware route choices: This version of DSR uses transmit power control and employs power aware route selection. The energy aware link cache is used without energy aware route discovery and maintenance.

- Energy Aware DSR with the energy aware link cache: This is the *minimum energy routing* version of the DSR using the energy aware link cache with energy aware route discovery and maintenance.

VIII. Test procedure

We used random way-point movement scenarios to study the effect of mobility on the minimum energy routing protocols. The average node speeds considered were 0.1 m/s, 1 m/s and 10 m/s. The number of nodes considered were 10 and topology size was 1000x300 m^2. Each simulation run for one particular node speed consisted of 11 random scenarios.

We also compared the performance of the Energy Aware DSR with path cache and Energy Aware DSR with the energy aware link cache in terms of scaling. The number of nodes considered were 10, 15, 20 and 25 with topology sizes as 600x300, 1000x300, 1200x300 and 1500x300 m^2 respectively.

The traffic sources considered were CBR sources with a packet rate of 1 packet/sec/node and packet size of 512 bytes. In the traffic model we use, each node remains idle for a time t_{idle} which is exponentially distributed with a mean of 30 seconds. It then sources a flow to a randomly selected destination node; the flow lifetime t_{flow} is exponentially distributed with a mean of 300 seconds. This repeats for each node till the end of the simulation at time t = 1500 seconds.

The results for the various protocols for different mobility speeds in the 10 nodes case can be observed in Fig. 6 – 8. Fig. 9 shows the effect of scaling on the performance of the minimum energy routing protocol versions.The error bars indicate the standard deviation on each segment of the stacked bar chart.

The results for different mobility rates indicate that the Energy Aware DSR protocol with the energy aware link cache performs best in static and slow mobility scenarios where its energy consumption is within a factor of 3 of the God energy. As the mobility of the scenario rises, the routing protocol's energy consumption rises above the factor of 3, but it still performs better than the existing protocol version in terms of energy savings. The importance of energy aware route discovery and route maintenance can be clearly observed from the additional energy savings achieved by the Energy Aware DSR protocol over the DSR with link cache with transmit power control and power aware route choices.

The results for different number of nodes in the network gives an indication regarding scaling issues of the protocol versions. From the results it is clear that the Energy Aware DSR with path cache will not scale as the routing overhead increases quickly with the number of nodes N. The routing overhead of the Energy Aware DSR with the energy aware link cache has a minimal increase with the number of nodes as

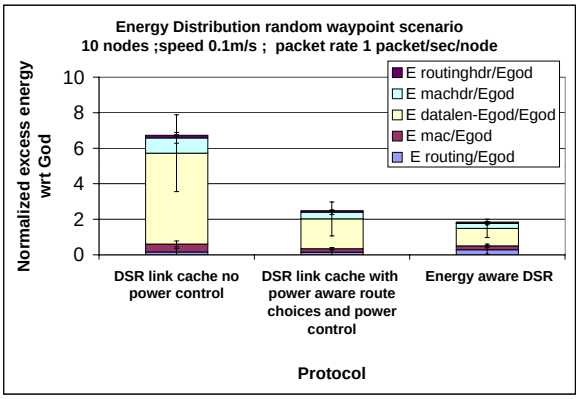


Figure 6: 10 node random way-point scenario speed 0.1 m/sec

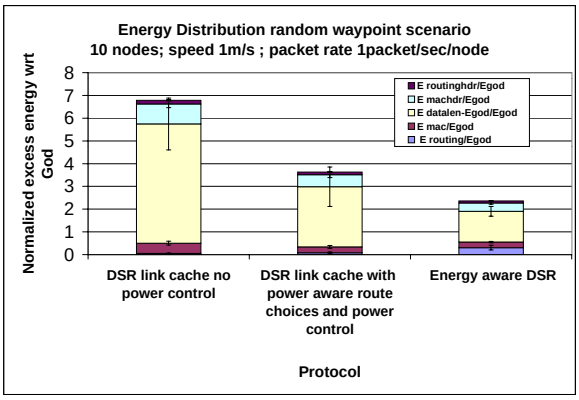


Figure 7: 10 node random way-point scenario speed 1 m/sec

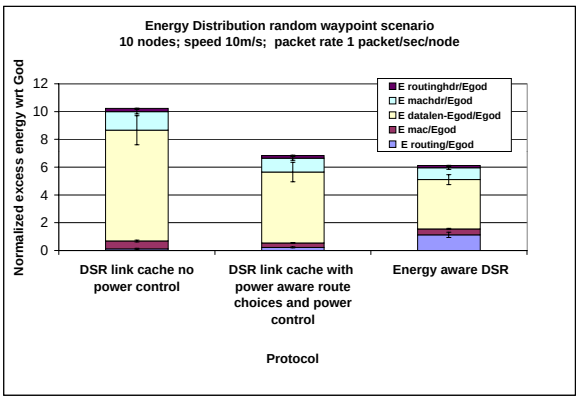


Figure 8: 10 nodes random way-point scenario speed 10 m/sec

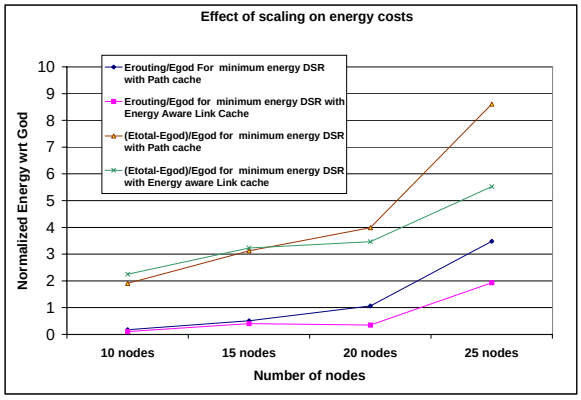


Figure 9: Effect of scaling on energy cost- Node speed 1 m/sec

seen in the results and thus this indicates that this protocol will scale better with the number of nodes in the network.

It can also be observed that the reduction in the energy consumption by using the minimum energy routing protocol is primarily due to the reduction in the routing inefficiency of the protocol and the use of transmit power control which is brought about by implementing the requirements of the minimum energy routing protocols.

IX. Implementation of the minimum energy routing protocol on a hardware test-bed

In this section we discuss the issues regarding the implementation of the minimum energy routing protocol on a hardware test-bed.

IX.A. Choice of Wireless Ethernet card

The minimum energy routing protocol employs dynamic transmit power control to send the packets out at the minimum transmit power. Hence it is imperative to choose a wireless Ethernet card that supports this feature. Among all the cards available that support this feature, the Cisco Aironet 350 series card [1] was chosen as it offers 6 levels of transmit power control. The available power levels are 100 mW (20 dBm), 50 mW (17 dBm), 30 mW (15 dBm), 20 mW (13 dBm), 5 mW (7 dBm) and 1 mW (0 dBm).

IX.B. Choice of Operating System and routing infrastructure

Among all operating systems, the obvious choice for the operating system for the test-bed is Linux. The Linux operating system allows applications running in user-space to tap off packets from the network stack using modules like EtherTap, process these packets and re-insert the packets back into the stack after modifying their contents. This allows for a layer of kernel independence in the protocol implementation which is much desired. Moreover, Linux being open source allows for easy modification of the device driver for the wireless Ethernet card to enable setting the transmit power dynamically.

The Click modular router [11] is chosen as the routing infrastructure to implement the protocol. The router can be configured to run at user level using a driver program or in the Linux kernel as a kernel module. When run in the user level, it requires a kernel tap that captures packets that are destined to or from the kernel. This allows the packets to be manipulated in the user-space and also allows for the re-insertion of the packets into the kernel stack. When run as a kernel module, it can steal packets from the network devices before Linux gets a chance to handle them; it sends packets directly to the devices and also send packets to Linux for normal processing.

Click provides an interconnected collection of modules called elements. Every element controls a specific aspect of the router. They handle the functions of communicating with the network devices, packet modifications and packet scheduling. These elements are inter-connected to create the router configuration file. The Click program reads the configuration file, and sets up the router accordingly. One instance of a successful implementation of an ad hoc routing protocol on Click is the AODV implementation done by Tornquist et. al. [25].

IX.C. Implementing dynamic transmit power control

Before the packets are transmitted, the minimum energy routing protocol should be able to set the transmit power of the card to the minimum transmit power value at which the packet is supposed to be sent. The iwconfig tool [26] in Linux allows for setting the transmit power value of the wireless Ethernet card dynamically (for cards that support this feature).

The command

```
iwconfig ethX txpower N
```

sets the transmit power of the card on the interface ethX to N dBm. This is done by an ioctl call to the card's driver with the appropriate parameters.

The cards run on the Aironet driver that is bundled with the PCMCIA Card services package available from http://pcmcia-cs.sourceforge.net. The driver code was modified so that it supported the dynamic setting of the transmit power.

IX.D. Obtaining Received signal strength of a packet

To compute the minimum transmit power value for a link, the mechanism suggested in Section VI.B needs the value of P_{RX} which is the received signal strength of the packet. The iwspy tool [26] can be used to obtain this received signal strength. The iwspy tool allows the user to set a list of network addresses in the driver. The driver will gather quality information for each of those addresses (updated each time it receives a packet from that address). The tool allows the user to display the information associated with each address in the list. Iwspy accepts an IP address as well as hardware addresses (MAC). IP addresses will be converted to hardware addresses before being transmitted to the driver.

The output obtained by the iwspy tool typically looks like:

```
08:00:0E:21:D7:4E:Quality 15;Signal 29;
Noise 0(updated)
08:00:0E:21:3A:1F:Quality 0;Signal 0;
Noise 0
08:00:0E:2A:26:FA:Quality 0;Signal 0;
Noise 0
```

The (updated) flag indicates that the statistics have been updated for the latest packet that has been received from the specified address. The received signal strength of a packet transmitted by a specific sender can be easily extracted from the 'Signal' information obtained using the iwspy tool.

In the following sections we discuss in detail the design and implementation of the Click DSR router and the Click Energy Aware DSR router.

X. Click DSR router implementation

This section describes our implementation of DSR using Click. It provides a basis for describing our energy aware modifications to the protocol in the next section. The user-level mode of Click is used to implement the DSR router. The DSR protocol is defined in the IETF Draft and is comprehensive in stating the exact implementation details of the protocol. We have conformed to the draft while designing the Click DSR router elements. In the following subsections we discuss the DSR packet structure and the Click elements for the DSR router.

X.A. DSR packet structure

This section enumerates the structure of the packets used in the DSR protocol. The individual header format conforms to the IETF draft.

The following types of DSR packets are defined:

- Data packet: The data packet consists of an IP header followed by the DSR headers. The DSR fixed header stores the length of all the DSR headers. The DSR source route header is followed by the entire source route of the packet that essentially is a list of addresses of the intermediate nodes. The DSR ACK request header requests the next hop to send back an ACK packet that is required for link maintenance. The data portion of the packet follows the ACK request header.
- Route Request packet: The Route Request packet consists of an IP header followed by the DSR fixed header. The DSR Route Request header and the list of addresses of the intermediate nodes that have forwarded the route request follow the DSR fixed header.

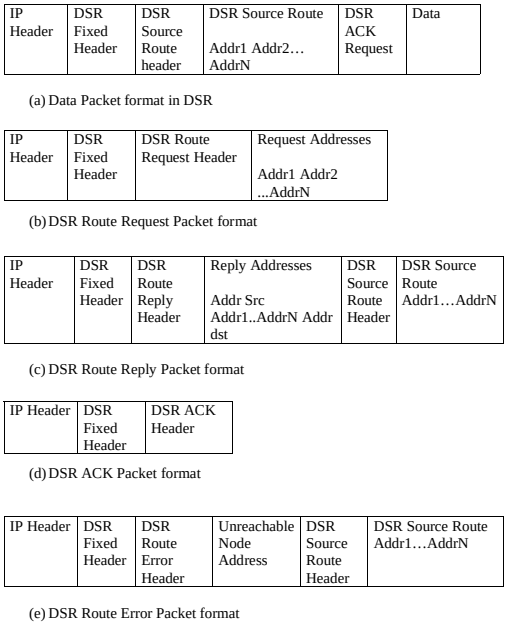


Figure 10: Packet formats in the Click DSR implementation

- Route Reply packet: The route reply packet consists of the IP header followed by the fixed DSR header. The route reply header follows the fixed DSR header. This is followed by the reply route, which is a list of addresses from the source to the destination. The source route header for the route reply packet is then included along with the node addresses.
- ACK packet: The DSR ACK packet consists of the IP header followed by the DSR fixed header

and the DSR ACK header. The DSR ACK packet is an optional packet but necessary in our energy aware implementation in order to bring about network level route maintenance.

- Route Error packet: The Route Error packet consists of the IP header, fixed DSR header, Route Error header and finally the Unreachable Node Address, which is the address of the node at the end of the broken link. The source header and the source route for this route error packet then follow.

X.B. Click elements for DSR Router

This subsection discusses the design of the Click DSR Router elements. The DSR specific elements are:

- RouteTable with SendBuffer: The RouteTable element is responsible for the basic routing that takes place in the DSR protocol. This element includes the cache for saving the routes obtained by a node during route discovery. A link cache is used to cache these routes. An entry in the link cache is of the form (Node From, Node To, Cost) where Node From is the source of the link and Node To is the destination of the link. Cost of the link is the hop count, which in this case is one. When a route is looked up in the cache, the minimum cost route is selected using Dijkstra's algorithm.

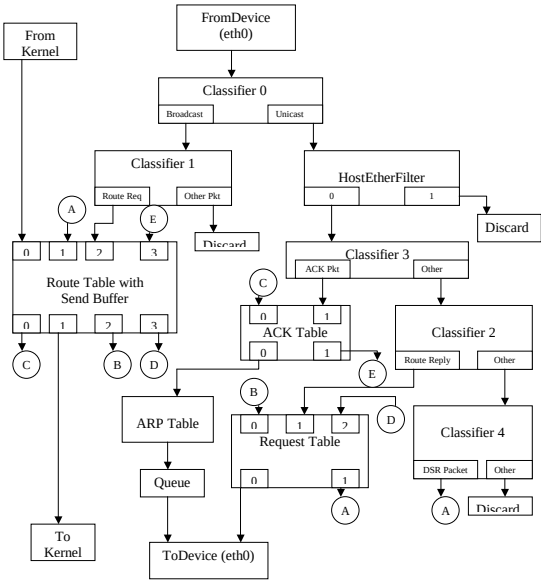


Figure 11: The Click DSR router

A packet with an unknown destination route is saved in a FIFO buffer called SendBuffer. This buffer has a time stamp associated with every packet. The buffer is probed every second and route requests are initiated for packets in the buffer whose destination routes have not yet been found but whose time out values have not expired. The packets residing in the buffer with expired time out values are discarded. The element also handles incoming data packets, route replies and route errors that need to be forwarded to the final destination. In case the link to the next hop for a packet becomes unavailable, the element removes that particular link from the cache and is responsible for sending route error packets to the source node. Route Error packets from other nodes are processed and the broken link is removed from the link cache. The element is also responsible for sending out ACK packets to the source of the ACK request.

- ACKTable: This element is responsible for link maintenance. ACK Request entries are stored in the ACKTable along with the clone of the data packet that is sent out. If the ACK packet is not received within a certain timeout period, the link is declared as invalid and the packet is forwarded to a particular output of the router. This output is connected to an input of the RouteTable element and the link is removed from the cache. If the ACK packet is received within that timeout period, that ACK entry is removed from the ACKTable.

- RequestTable: The RequestTable element is responsible for sending out route request packets to find the route to a particular destination. It also forwards route requests from other source nodes after adding itself in the route request header. A back off algorithm is employed to control the number of route requests that are sent out for a particular source destination pair. After a route reply is obtained, the request table removes that particular request entry from the table.

- ARPTable: This element is responsible for looking up the MAC Address for a given IP Address. This is carried out by looking up a mapping table that contains these entries. On a successful lookup, the element adds an ethernet header to the incoming packet and forwards it on to an output port.

The elements are connected together in a configuration file as shown in Fig 11 to make up the DSR router. This This router configuration runs on all the nodes to form the DSR ad hoc network.

XI. Energy Aware DSR router implementation

IP Header	DSR Fixed Header	DSR Source header	DSR Source Route Addr1 Addr2..AddrN	DSR Source Powers ToPwr1… ToPwrN	DSR ACK Request	Data

(a) Data Packet format in Energy Aware DSR

IP Header	DSR Fixed Header	DSR Route Request Header	Request Addresses Addr1 Addr2..AddrN	Power Values ToPwr1…ToPwrN

(b) Energy Aware DSR Route Request Packet format

IP Header	DSR Fixed Header	DSR Route Reply Header	Reply Addresses Addr Src Addr1.. AddrN.. Addr Dst	Reply Powers ToPwr Src ToPwr1 … ToPwrN ToPwr Dst	DSR Source Route Header	DSR Source Route Addr1… AddrN	DSR Source Route Powers ToPwr1.. To PwrN

(c) Energy Aware DSR Route Reply Packet format

IP Header	DSR Fixed Header	DSR ACK Header

(d) Energy Aware DSR ACK Packet format

IP Header	DSR Fixed Header	DSR Route Error Header	Unreachable Node Address	DSR Source Route Header	DSR Source Route Addr1…AddrN	DSR Source Route Powers ToPwr1..ToPwrN

(e) Energy Aware DSR Route Error Packet format

Figure 12: Packet formats for Energy Aware DSR implementation

In this section we enumerate the changes to be made to the DSR router discussed in the previous section to make the protocol energy aware. The energy aware implementation involves the usage of an energy aware routing metric. The cost of each link in the route is computed in terms of power, which translates to energy cost of the link. The metric thus is minimum transmit power required for successful communication for a given link. To make the protocol energy aware, these power values need to be advertised during the route discovery and stored in the link cache as the cost of the links. We achieve this by adding a power header to the original DSR packet structure. The header is a list of power values that are the minimum transmit powers for each link. The computation of these power values has been discussed in Section VI.A. The inclusion of the power aware header will lead to making changes in the format of the DSR packets discussed in the previous section. In the following subsection we detail these changes.

XI.A. Changes to the Packet Structure

The packet formats are shown in Fig.12. The packets are similar to the DST formats with the following changes:

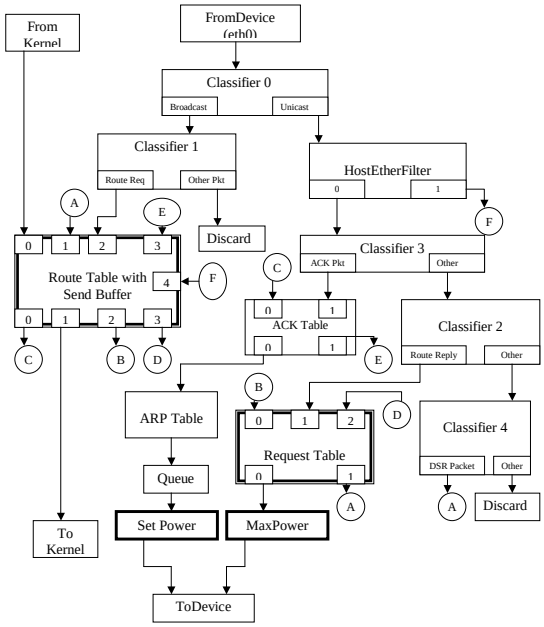


Figure 13: Energy Aware DSR implementation in Click

- Data packet: The data packets are forwarded to the next hop at transmit power given by the DSR Source Power values.
- Route request packet: The power values are added to the original Route Request packet. Topwr1 refers to the power required to get to Addr1 from the previous hop. The route request packets are transmitted at full power i.e. 20dBm for the Cisco Aironet 350 Cards.
- Route Reply packet: The reply path powers and the source path powers are included in the new route reply packet format. The route replies are forwarded to the next hop as per the source route at transmit powers given by the DSR Source Power values.
- ACK packet: ACKs are currently sent out at full power i.e. 20dBm for Cisco Aironet 350 Cards.
- Route Error packet: No change.

XI.B. Addition of new elements

Two additional elements were added to the original implementation of Click DSR Router. They were added to carry out transmit power control. These two elements use the iwconfig call that is discussed in Section IX.C.

- Set Power: This element carries out the functionality of setting the transmit power of the packet to the minimum transmit power value.

- Max Power: This element carries out the functionality of setting the transmit power to the maximum value, which is 20dBm for the Cisco Aironet 350 Cards.

XI.C. Changes in the existing elements

Two existing elements were changed. The RouteTable with SendBuffer has four modifications

1. Minimum Transmit Power Computation: Computing the minimum transmit power requires the computation of the received signal strength. To compute the received signal strength of the packet received from the transmitter node, ioctl calls are used. The MAC Address of the transmitter node is passed as a parameter to this ioctl call. This call returns the received signal strength [1] of the packet that is transmitted by the transmitter node.

   Minimum Transmit Power is then computed using the RSSI (dBm) value using (12).

2. Addition of Power Header: The element needs to include the power header while forming the different types of Energy Aware DSR packets. This involves changes in the functions that add the headers to the packets.

3. Extraction of Power Header: The element needs to read the power values from the received packets to add the information to the link cache.

4. Energy Aware route maintenance: An extra input port is added to the element that processes snooped packets and discovers new minimum energy routes. The element then sends out a gratuitous route reply to the source of the packet. Energy aware route maintenance also involves constantly monitoring the power cost of each link in the route. This involves changing the DSR source header to include an additional flag that indicates a change in the power costs of the links in the route. The destination node checks this flag and sends out a gratuitous reply to the source informing it about the new costs of the route.

The RequestTable element is also modified to implement minimum transmit power computation to calculate the minimum transmit power value to the previous node. This value is then added to the power list in the route request packet when it has to be rebroadcast. Thus the RequestTable element now propagates the new energy metric through the network.

Summarizing, the DSR protocol has been implemented on a hardware test-bed using the Click modular router. We have also incorporated the modifications to make this protocol energy aware. The minimum energy route is used by the energy aware DSR router to route the packets. These packets are transmitted on the medium using transmit power control. In our lab we have demonstrated each of the energy aware mechanisms on our hardware test-bed. Work is on to quantify the energy savings obtained by the use of the energy aware DSR protocol.

XII. Conclusions

Our experimental results justify the necessity of the required features of a minimum energy routing protocol. These features are very generic in nature and can be used as a guideline for designing new minimum energy routing protocols. We have shown that these features can be easily implemented on an existing version of the protocol to derive a 'minimum energy routing' version of the protocol. The minimum energy routing protocol energy consumption is around a factor of three times more than the God energy at low speeds as compared to eight times more in the case of the existing protocol. At higher speeds the ratios are seven and eleven. It is also clear from the results that a minimum energy routing protocol should employ a unified link cache graph data structure to store the routing and power information so that it can converge to the minimum energy route faster with reduced routing overhead. The minimum energy routing protocol has been implemented on laptops running Linux using wireless Ethernet cards that support the transmit power control feature. The Click modular router infrastructure has been used to develop the protocol in the form of modules. Work is going on to quantify the energy savings that can be achieved by using the minimum energy routing protocol in a real life testbed environment and comparing them to the results obtained from the ns simulations.

[1] For the Cisco Aironet 350 cards, the RSSI value has a special percentage format that is converted to dBm using

$$RSSI(dBm) = RSSI(percentage)/2 - 95 \qquad (15)$$

References

[1] Cisco Systems, "Cisco Aironet 350 Series Client Adapter Data Sheets"

[2] Broch, J., Maltz, D., et al. “A performance comparison of multi-hop wireless ad hoc network routing protocols,” *Proc. of MOBICOM*, 1998, pp. 85–97.

[3] Brown, T.X, Doshi, S., Zhang, Q., “Optimal power aware routing in a wireless ad hoc network,” *IEEE LANMAN 2001 Workshop Proceedings*, pp. 102–105.

[4] Brown, T.X, Zhang, Q., Gabow, H., “Maximum Flow Life Curve for a Wireless Ad Hoc Network,” *MOBIHOC 2001*

[5] Chang, J., Tassiluas, L., “Energy Conserving Routing in Wireless Ad-Hoc Networks,” *Proceedings of IEEE INFOCOM 2000*, pp. 22-31, 2000.

[6] Chen, B., Jamieson, K., Balakrishnan, H., Morris, R., “Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks”, *Proceedings of MOBICOM 2001*, 2001.

[7] Fall, K., Varadhan, K., “The ns Manual,”*(formerly ns Notes and Documentation)*, The VINT Project: A collaboration between researchers at UC Berkeley, LBL, USC/ISI and Xerox PARC

[8] Feeney, L., Nilsson, M., “Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment,” *IEEE INFOCOM 2001*

[9] Grace, D., Tozer, T., Burr, A., “Reducing Call Dropping in Distributed Dynamic Channel Assignment Algorithms by Incorporating Power Control in Wireless Ad Hoc Networks,” *IEEE JSAC*, Vol. 18, No. 11, November 2000, pp. 2417–2428.

[10] Hu, Y., Johnson, D., “Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks,” *Proceedings of MobiCom 2000*, August 2000, pp. 231-242.

[11] Kohler, E., *The Click modular router*, Ph.D. thesis, Massachusetts Institute of Technology, February 2001.

[12] Li, Q., Aslam, J., Rus, D., “Online Power-aware Routing in Wireless Ad-Hoc Networks”, *Proceedings of MOBICOM 2001*, 2001.

[13] IEEE Standards Department, “Draft Standard IEEE 802.11 Wireless LANs”.

[14] Johnson, D., Maltz, D., “Dynamic Source Routing in Ad Hoc Wireless Networks,” *Mobile Computing*, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[15] Maltz, D., Broch, J., Jetcheva, J., Johnson, D. “The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks,” *IEEE JSAC*, August 1999, Volume 17, Number 8, pp. 1439-1453

[16] Maltz, D., Broch, J., Johnson, D., “Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed,” *CMU School of Computer Science Technical Report CMU-CS-99-116*, March 1999.

[17] Perkins, C., Bhagwat, P., “Highly Dynamic Desination-Sequenced-Distance-Vector Routing (DSDV) for Mobile Computers,” *Comp. Commun. Rev.*, Oct. 1994, pp. 234-44.

[18] Perkins, C., Royer, E., “Ad-Hoc On-Demand Distance Vector Routing,” *Proc. 2nd IEEE Wksp. Mobile Comp. Sys. and Apps.*, Feb. 1999, pp. 90-100.

[19] Ramanathan, R., Rosales-Hain, R., “Topology control of Multihop Wireless Networks using transmit power adjustment,” *IEEE INFOCOM 2000*, pp. 404–413.

[20] Roduplu, V., Meng, T., “Minimum energy mobile wireless networks,” *IEEE JSAC*, v. 17, n. 8, Aug. 1999, pp. 1333-44.

[21] Royer, E., Toh, C., “A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks,” *IEEE Personal Communications*, April 1999, pp. 46–55.

[22] Shih, E., Cho, S., Ickes, N., Min, R., Sinha, A., Wang, A., Chandrakasan, A., ‘Physical-Layer Driven Protocol and Algorithm Design for Enery-Efficient Wireless Sensor Networks”, *Proceedings of MOBICOM 2001*, 2001.

[23] Singh, S., Woo, M., and Raghavendra, C.S., “Power aware routing in mobile ad hoc networks,” *Proc. of MOBICOM*, 1998, pp. 181–190.

[24] Subbarao, M.W., "Dynamic Power-Conscious Routing for MANETs: An Initial Approach," *Proc. IEEE VTC*, Amsterdam, The Netherlands, Sept. 1999.

[25] Tornquist A., Neufeld M., Grunwald D., "The Design of a Modular Implementation of Ad Hoc Routing Protocols," *Submitted to INFOCOM 2001*

[26] Tourrilhes, J., "The Linux Wireless LAN Howto" $www.hpl.hp.com/personal/Jean\_Tourrilhes/Linux$, accessed on 30th September 2001

[27] Xu, Y., Heidemann, J., Estrin, D., 'Geography-informed Energy Conservation for Ad Hoc Routing" *Proceedings of MOBICOM 2001*, 2001.

Biographies

Sheetalkumar Doshi is a graduate student in Electrical Engineering at the University of Colorado at Boulder. He completed his Bachelor's degree in Electronics and Telecommunications engineering from University of Pune, Pune, India. His research interests are in wireless ad hoc networks and mobile computing. He can be reached at doshi@colorado.edu.

Shweta Bhandare is a graduate student in Interdisciplinary Telecommunications at the University of Colorado at Boulder. She completed her Bachelor's degree in Computer Science and Engineering from Goa University, Goa, India. Her research interests are in wireless ad hoc networks. She can be reached at bhandare@colorado.edu.

Timothy X Brown is an associate professor at the University of Colorado, Boulder. He received his B.S. in physics from Pennsylvania State University and his Ph.D. in electrical engineering from California Institute of Technology in 1990 when he joined the Jet Propulsion Lab. In 1992 he joined Bell Communications Research. Since 1995 he has been jointly in Electrical Engineering and Interdisciplinary Telecommunications at the University of Colorado, Boulder. His research interests include adaptive network control, machine learning, and wireless communications systems. He is a recipient of the NSF CAREER Award. He can be reached at timxb@colorado.edu.