

Stability Aware Routing: Exploiting Transient Route Availability in MANETs

Pramita Mitra¹, Christian Poellabauer¹ and Shivajit Mohapatra²

¹ Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556

² Applications Research Center, Motorola Labs, 1295 E. Algonquin Road, MD: 2nd floor, Schaumburg, IL 60196

Abstract. The highly dynamic character of a mobile ad-hoc network (MANET) poses significant challenges on network communications and resource management. Previous work on routing in MANETs has resulted in numerous routing protocols that aim at satisfying constraints such as minimum hop/distance or low energy. Existing routing protocols often fail to discover stable routes between source and sink when route availability is transient, e.g., due to mobile devices switching their network cards into low-power sleep modes whenever no communication is taking place. In this paper, we introduce a stability aware dynamic source routing protocol (SA-DSR) that is capable of predicting the stability (i.e., expiration time) of multiple routes. SA-DSR then selects the route that minimizes hop count while staying available for the expected duration of packet transmission. Comparisons of SA-DSR to the original DSR (Dynamic Source Routing) protocol indicate a significant (up to 31%) increase in successful packet transmissions with comparable route establishment and maintenance overheads.

Keywords

Mobile Ad Hoc Networks (MANET), Dynamic Source Routing (DSR), Dynamic Power Management (DPM), Stability Aware Routing

1 Introduction

Conventional MANET routing protocols do not consider power as a design constraint, instead, they tend to search for optimal routes in terms of delay. In such algorithms, connection between two nodes is established through nodes on the shortest path routes, which may however result in quick depletion of the battery of the nodes along the most heavily used routes in the network and eventual network partition and low connectivity.

Dynamic Power Management (DPM) in MANETs has gained huge popularity over the last decade. Mobile nodes in wireless ad-hoc networks often put their wireless network cards to sleep when they are not transmitting or receiving data. In most MANETs, wireless traffic is infrequent and recent work [1] shows that wireless network

cards should be turned inactive for 50% or less of the entire lifetime to obtain a balance between optimal power-saving and sustained network connectivity. But sleep modes can lead to loss of network connectivity and hence lower the packet delivery ratio.

This paper³ focuses on routing in MANETs with transient route availabilities, i.e., route establishment takes into consideration the expiration time and therefore the stability of a potential route. This approach, called *Stability Aware Dynamic Source Routing (SA-DSR)*, is based on the prediction of future sleep times of mobile nodes (i.e., the times when mobile nodes' DPM techniques will turn off their network cards). The goal of this approach is to introduce DPM-awareness in routing decisions and thereby to increase the number of successful packet transmissions. While SA-DSR is an extension to the well-known on-demand Dynamic Source Routing (DSR) [10], the concept of stability awareness can be added to any routing protocol. A variety of *stability predictors* can be used, including *hints* given by applications and/or the DPM mechanism. In this paper, SA-DSR monitors the *packet scheduler queue* for the network driver and predicts the next down-time of the network card. That is, SA-DSR conservatively predicts that the network card will enter its next sleep mode after all packets in the queue have been transmitted plus a driver-specific timeout value.

The remainder of the paper is organized as follows: Section 2 compares SA-DSR to previous work. Section 3 discusses the system model for the proposed routing protocol. In Section 4, we describe the details of SA-DSR, followed by simulation results in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

In [2], Chin et al. present their experience of implementing two popular MANET protocols and report that due to fading and transient network links both of these protocols fail to offer a stable route over any multi-hop network connection. They also suggest the need for using a route stability metric during the route discovery phase of MANET routing protocols. This observation has given rise to a number of algorithms that address energy management and routing in wireless ad-hoc networks. We broadly classify these algorithms into three categories: *Transmit Power Aware Routing*, *Residual Energy Aware Routing* and *Network Sleep Time-Aware Routing*. This section compares our SA-DSR protocol to these related categories.

2.1 Transmit Power Aware Routing

Such algorithms minimize the transmit power required for packet transmission or adjust the transmit power of nodes with varying network traffic and remaining node energy. In [3], Toh et al. propose the conditional max-min battery capacity routing algorithm which chooses the route with minimal total transmission power if all nodes in the route have remaining battery capacities higher than a threshold; otherwise, routes that consist of nodes with the lowest remaining battery capacities are avoided. In [4], Tarique et al. integrate two common energy management approaches: they use

³ This work was made possible by NSF grant 0545899.

a load sharing approach for routing decisions and a transmit power control approach for link by link power adjustments. They employ their approach to enhance DSR [10].

Such algorithms in general select the minimum transmit power cost routes. Though some of them take the node residual energy into account, but mostly nodes along the least transmit power cost routes tend to die soon since these routes now become the most heavily used ones instead of the min-hop ones. This is harmful since the nodes which die early are precisely the ones that are needed most to maintain network connectivity. The SA-DSR protocol does find the optimal route not only based on a metric like min-hop, but also a second metric (reliability). It finds multiple stable routes for a particular pair of source and sink nodes and thus maintains the network connectivity.

2.2 Residual Energy Aware Routing

Such algorithms minimize the residual energy of the nodes and select the most residual energy or least battery cost routes. In [5], Marbukh et al. aim at preserving network connectivity by choosing routes according to the remaining battery life of nodes along the route. They use a power draining factor to accurately predict the residual battery life time. In [6], Venugopal et al. study various ad-hoc network protocols in terms of robustness and conclude that the robustness of a routing protocol is restricted by its remaining energy. Further, they present a Max-Min Energy DSR (MME-DSR) route selection algorithm to select the optimal energy route. In [7], Maleki et al. propose a lifetime prediction routing protocol for MANETs that maximizes the network lifetime by selecting routes that minimize the variance of the residual energies of the nodes in the network and include the rate of energy discharge into the cost function to improve network lifetime. They argue that mobility of nodes can affect the traffic pattern through the nodes and the recent history is a good indicator of this traffic.

These works assume that it is better to use a higher transmit-power cost route if the least transmit-power cost route consists of nodes with small amount of residual energy. Nodes usually do idle listening when there is no significant traffic. Such algorithms never completely turn off the nodes in absence of traffic. SA-DSR realizes a better approach for power-saving by utilizing a Dynamic Power Management (DPM) module that puts wireless nodes into a sleep mode when the node is not transmitting or receiving data. It finds stable routes as predicted by a DPM-aware *Route Stability Prediction Algorithm* and thus ensures acceptable network connectivity.

2.3 Network Sleep Time Aware Routing

In [8], Chia et al. propose that devices which are not currently active in any data communication may enter a sleep state, but can be powered up remotely through a signal using a simple circuit based on RF technology. Radio devices select different time-out values (*sleep patterns*), to enter various sleep states depending on their battery status and quality of service. In [9], Singh et al. employ a MAC layer protocol for PAMAS (Power Aware Multiple Access protocol with Signaling) in which nodes overhearing transmissions between two other nodes turn themselves off and wake up after an interval of time equal to the total transmission time as indicated in the

RTS/CTS message exchange between the sender-receiver pairs. They deploy metrics such as *minimize energy consumed per packet* or *minimize time to network partition*, and verify these metrics with their proposed MAC layer protocol.

In our SA-DSR protocol, the sleep and awake schedule is determined from prediction of link expiration based on the queue contents of the packet scheduler and the network interface device timeout value. The DPM schedule is somewhat conservative since it ignores the possibility of more packets being added before the timeout expires.

3 System Model

DPM supports energy conservation by making mobile nodes put their wireless network cards to sleep when no data communication is taking place. A consequence of this technique is that mobile nodes will be unreachable for large periods of time. Therefore, we need to know the accurate network ‘up’ and ‘down’ times (DPM schedule) in order to introduce DPM awareness in routing decisions. Currently SA-DSR predicts the DPM schedule for mobile nodes from the queue contents of the packet scheduler and the network device timeout value. Toward that end, the SA-DSR protocol computes the minimum time to transmit all packets currently residing in the packet scheduling queue and adds the device-specific timeout value, i.e.: $t_{exp} = n * t_{send} + t_{timeout}$, where n is the number of bytes to be sent, t_{send} is the minimum time needed to transmit one byte over the medium, and $t_{timeout}$ is the amount of time the network card stays in active mode after the last transmission.

We define the *route uptime factor* (RUF) as a metric which indicates the *next earliest time* when the link between any pair of adjacent nodes on a route is going to be interrupted due to one (or both) of the nodes being put to sleep. Now we derive RUF as follows:

If we assume nodes as vertices and the links between the nodes as edges connecting them, then let $G(V,E)$ be the graph representing the network topology where V is the set of vertices and E is the set of edges. Let

$$R_{ij} = (V_i, V_{i+1}, V_{i+2}, \dots, V_K, V_{K+1}, \dots, V_{j-1}, V_j) \quad (1)$$

be the route from source node V_i to V_j through intermediate nodes V_k, V_{k+1} , etc. Let Ω_{ij} be the set of all possible alive routes between V_i and V_j . The DPM sleeping schedule S_{ij} for the route R_{ij} is defined as

$$S_{ij} = (t_i^{off}, t_{i+1}^{off}, t_{i+2}^{off}, \dots, t_K^{off}, t_{K+1}^{off}, \dots, t_{j-1}^{off}, t_j^{off}) \quad (2)$$

where t_i^{off} is the next earliest time to sleep for node V_i . We define the *link uptime vector* or L_{ij} for the route R_{ij} as

$$L_{ij} = (t_i^{uptime}, t_{i+1}^{uptime}, t_{i+2}^{uptime}, \dots, t_K^{uptime}, t_{K+1}^{uptime}, \dots, t_{j-1}^{uptime}) \quad (3)$$

where t_i^{uptime} is the uptime of the link $E_{i,i+1}$ connecting nodes V_i and V_{i+1} and is defined by $\min(t_i^{off}, t_{i+1}^{off})$, since uptime of a link is determined by how long the

link will be alive before breaking down due to one of its end nodes going to sleep and thus essentially is expressed by the minimum of the DPM sleeping schedule of the end nodes. The *route uptime factor* RUF_{ij} for route R_{ij} can be expressed as the minimum of the link uptime vector L_{ij} along the route since it will indicate how long the route will be alive before breaking down due to the break in any of its constituent links:

$$RUF_{ij} = \min(t_i^{uptime}, V_i \in V, t_i^{uptime} \in L_{ij}) \quad (4)$$

Therefore, we can summarize the problem as obtained in Problem 1.

Problem 1 *Given the next earliest time to sleep t_i^{off} for each node $V_i \in V$ in the graph $G(V,E)$, accumulate the set of all possible routes Ω_{ij} between nodes V_i and V_j with the corresponding route uptime factors RUF_{ij} for each $R_{ij} \in \Omega_{ij}$ and find the min-hop route R_{ij} from the set of all stable routes Ω_{ij} . If there are more than one routes with the same min-hop length, then the one with the maximum route uptime factor value is selected.*

4 Stability Aware Dynamic Source Routing

There are two key differences between SA-DSR and standard DSR. Firstly, SA-DSR introduces *DPM Awareness in routing decisions* at intermediate nodes. The discovered routes are always stable since the routing module employs a *Route Stability Prediction (RSP) Algorithm* (Algorithm 1) and proactively discards the forwarding of RREQ packets for predicted unstable routes, on the basis of the Link Uptime Vector L_{ij} contained in the RREQ packet. L_{ij} is dynamically formed by intermediate nodes during the route discovery phase. Secondly, unlike DSR, SA-DSR finds *multiple stable routes* to the target node by allowing the intermediate nodes to rebroadcast multiple RREQ packets with the same <source address, request id> pair containing distinct source routes. Although having more route discovery can contribute to an increase in the total network traffic, this increase in traffic is expected to be compensated by proactively avoiding the forwarding of RREQ packets for predicted unstable routes by intermediate nodes.

SA-DSR consists of three steps which will be described in the remainder of this section; (1) Route discovery phase, (2) Route reply phase, and (3) Route selection phase in the source node.

4.1 Route discovery phase

When a source node needs to send a data packet to a target node, it first searches its routing table for any entry using the target node address as the key. An entry in the routing cache contains a list of stable routes to the target node. If a routing cache entry is found, then the source node picks a route based on the *Route Selection (RS) Algorithm* (Algorithm 3). If no entry is found for the target node, then the source node initiates a route discovery for the target.

SA-DSR adds five new entry types to the RREQ packet format of standard DSR.

- Link Uptime Vector ($t_i^{uptime}, i \in (1, \dots, N - 1)$) for the route,
- DPM Sleeping Schedule for the last upstream node (t_{upstrm}^{off}),
- Timestamp at source node (T_{stamp}^{src}),
- Timestamp at last upstream node (T_{stamp}^{upstrm}) and,
- Per Hop Transmission Time (T_{trans}^{perhop}).

SA-DSR allows intermediate nodes to forward multiple RREQ packets with the same <source address, request id> pair if the packets contain distinct source routes. During the RREQ lookup at intermediate nodes, the 4-tuple <source address, request id, last upstream node address, partial route length> is checked with each entry in the recently seen requests list for possible match. If no match is found, then the RREQ contains a distinct source route and is eligible to be forwarded if the contained source route is predicted to be stable.

Route Stability Prediction (RSP) Algorithm: SA-DSR predicts the route stability using a link by link stability prediction. Each intermediate node executes the RSP algorithm for each received RREQ and predicts the stability of the link between itself and the last upstream node. All previous links in the source route are assumed to be stable, otherwise the previous upstream nodes would not have forwarded the RREQ packet. Thus the stability of the current link ensures the stability of the entire source route. For each received RREQ, the RSP algorithm in the $K + 1$ -th intermediate node V_{K+1} calculates the uptime of the link between itself and the last upstream node recorded in the RREQ and appends it to the Link Uptime Vector in the RREQ. It also calculates a running average of the per hop transmission time from the source node to itself T_{trans}^{perhop} , which takes into account varying per hop latency in case of the mobile nodes having various types of wireless interface devices (i.e., 802.11 or Bluetooth). If the uptime is less than $T_{stamp}^{src} + (3 * K * T_{trans}^{perhop})$, then the link will not be stable for the entire period of 3-way exchanges of the RREQ, the following RREP and then the data packet. Hence the intermediate node discards the RREQ. The detailed RSP algorithm is described in Algorithm 1.

If the partial source route $R_{src, K+1}$ is predicted stable, then the intermediate node V_{K+1} rebroadcasts the augmented RREQ which has the following fields as modified:

- The route record list appended by its own address.
- The Link UptimeVector augmented by the uptime of the link ($E_{K, K+1}$)
- Its own next earliest time to sleep as the DPM Sleeping Schedule of the last upstream node.
- The calculated running average Per Hop Transmission Time from the source node to itself as the Per Hop Transmission Time.
- Its current time as the new TimeStamp at the last upstream node.

The intermediate node V_{K+1} stores the <source address, request id, last upstream node address, partial route length> 4-tuple in its recently seen requests list.

4.2 Route reply phase

When the RREQ reaches the target node, it executes the RSP algorithm in the same way as the intermediate nodes. If the source route is predicted to be stable, the target

Algorithm 1: Route Stability Prediction (RSP) Algorithm at the local intermediate node V_{K+1}
<p>Input: received RREQ packet from last upstream node, next earliest time to sleep t_{K+1}^{off} for itself.</p> <p>Output: boolean true is route predicted alive, or false</p> <pre> 1 boolean alive:= true; 2 /*Compute Uptime t_K^{uptime} of the link $E_{K,K+1}$*/ 3 $t_K^{off} :=$ packet.getSleepingScheduleUpstreamNode(); 4 $t_K^{uptime} := \min(t_{K+1}^{off}, t_K^{off})$; 5 /*Compute Trans Time from Last Upstream Node to itself*/ 6 $T_{stamp}^{upstrm} :=$ packet.getTimeStampAtUpstreamNode(); 7 $T_{trans}^{perhop} :=$ packet.getPerHopTransTime(); 8 $T_{trans}^{upstrm} :=$ Current System Time - T_{stamp}^{upstrm}; 9 /*Compute Per Hop Trans Time from Source Node to itself*/ 10 $K :=$ packet.getNumLinkUptimeVector(); 11 $T_{trans}^{perhop} := ((K * T_{trans}^{perhop}) + T_{trans}^{upstrm}) / (K+1)$; 12 /*Predict if the route will be stable*/ 13 $T_{stamp}^{src} :=$ packet.getTimeStampAtSourceNode(); 14 $T_{transTot}^{src} := T_{stamp}^{src} + (3 * K * T_{trans}^{perhop})$; 15 if ($T_{transTot}^{src} > t_K^{uptime}$) 16 alive:= false; 17 return alive;</pre>

node sends an RREP packet back to the source along the reverse path recorded in the RREQ. SA-DSR adds three new entry types to the standard DSR RREP packet format:

1. *Link Uptime Vector:* The L_{ij} for the entire source route.
2. *Earliest Down Time:* T_{down}^{route} The minimum of all the Link Uptime Vector elements.
3. *Estimated Transmission Time:* $T_{transTot}^{src}$, as calculated in the route stability prediction phase.

The RSP algorithm during the route discovery phase did not take into account the length of the entire source route to the target node and hence made a prediction based on the stability of the partial route from the source node to itself. This might lead to some false predictions and hence needs to be corrected at the intermediate nodes during forwarding of the RREP packet. The intermediate nodes execute the *Modified Route Stability Prediction (MRSP) Algorithm* (Algorithm 2). Suppose there are total L links recorded in the source route of the RREQ and it arrives at the K -th intermediate node V_K (counting from the source node). V_K computes the time T_K^{more} at which the data packet will arrive at the target node after a two-way travel of total $(L + K - 1)$ number of links, since the RREP travels another $K - 1$ links to arrive at the source node and the data packet travels L links to arrive at the target node. If this T_K^{more} is less than T_{down}^{route} of the route, then V_K discards the RREP.

4.3 Route selection phase at the source node

A successfully received RREP in SA-DSR always ensures that the route is predicted to be stable for the estimated data transmission period (because of the conserva-

**Algorithm 2: Modified Route Stability
Prediction (MRSP) Algorithm**

```

Input: received RREP packet from last upstream node.
Output: boolean true is route predicted alive, or false
1 L:= packet.getNumLinkUptimeVector();
2 K:= packet.retNodeNum(localAddr);
3 /*Given the address of the current node, the above function
4 returns its node number in the source route*/
5  $T_{transTot}^{src} := \text{packet.getTransTime}();$ 
6  $T_{transPerHop} := T_{transTot}^{src}/L;$ 
7  $T_K^{more} := \text{current System Time} + (L + K - 1)*T_{transPerHop};$ 
8  $T_{down}^{route} := \text{packet.getEarliestDownTime}();$ 
9 if ( $T_K^{more} \leq T_{down}^{route}$ )
10 return true;
11 else
12 return false;

```

tive stability prediction used in our approach). A source node on having a successful RREP packet back, caches the learned route, the RUF for the route and estimated transmission time along the route in its routing table (the last two items are obtained from the *Earliest Down Time* and *Estimated Transmission Time* fields of the RREP respectively). The routes are stored in increasing order of their length in the routing table. When the source node sends a data packet, it exhaustively searches its routing table according to the *Route Selection (RS) Algorithm*, (Algorithm 3). The RS algorithm selects the min-hop stable route (the first entry in the routing table) for the target node. If there are more than one entry with the same min-hop length, then the route with the maximum value of RUF is chosen, since it has the highest predicted lifetime.

5 Experimental Results

In order to evaluate the performance of SA-DSR, we performed simulations to study large and complex network topologies. We chose the JiST/SWANS simulation environment. JiST⁴, Java in Simulation Time, is a discrete event simulator designed to run over a standard Java virtual machine. SWANS, a Scalable Wireless Ad hoc Network Simulator, is built on top of the JiST platform to provide the tools needed to construct a wireless mobile ad hoc network.

5.1 Simulation Setup

- The simulation driver gives us one sink node and a variable number of transmitting source nodes. SA-DSR will be implemented in a distributed event-based publish subscribe system where there will be only one event broker to match and deliver the events to the corresponding customers. The reason to use only one sink node is that the customers should be able to find stable routes to the single broker node.

⁴ <http://jist.ece.cornell.edu/>

Algorithm 3: Route Selection (RS) Algorithm

```
Input: routing table and the target node address
Output: the min-hop route from the set of stable routes.
1 /*initialize pointer to the first entry of routing table*/
2 rt_ptr:= routingTable.returnEntry(targetAddr);
3 min_hop:= MAX_HOPCOUNT;
4 route:= rt_ptr->entry;
5 /*extract the min_hop stable route from the set of stable routes and
6 if more than one entries with same length then extract the max-RUF one*/
7 while (rt_ptr){
8   if (rt_ptr->entry.hopCount == min_hop){
9     minRUFvalue:= min(route.RUFvalue , rt_ptr->entry.RUFvalue);
10    if (minRUFvalue == rt_ptr->entry.RUFvalue)
11      route:= rt_ptr->entry;
12  }
13  if (rt_ptr->entry.hopCount < min_hop){
14    min_hop:= entry.hopCount;
15    route:= rt_ptr->entry;
16  }
17  rt_ptr:= rt_ptr->next;
18 }
19 currentRUFvalue:= route.RUFvalue;
20 totalTransTime:= current System Time + route.TransTime;
21 if (totalTransTime < currentRUFvalue)
22   return route;
23 else
24   return NULL;
```

- The simulated network area is 2500mX2500m with 100 source nodes. Each transmitting source node has a bandwidth of 1 Mb/s and attempts to transmit one data packet of length 40 bits to the sink node.
- The simulation driver simulates a random interval repeatedly for each transmitting node. Each node stays awake for the input awake percentage of the simulated random interval and notifies the routing module about its next earliest time to sleep each time when it awakes.
- Each simulation was run for 5 trials, with a full range of awake percentages from 0% to 100% with an interval of 10%. Each trial was run for 500 seconds. Each transmitting source node attempts to send one data packet to the sink node.
- Nodes intermittently turn off their network cards for power-saving in absence of significant communication, which leads to link failures and low network connectivity. SA-DSR does not consider mobility as the cause of link breakage. The nodes used in the simulation are therefore stationary.

5.2 Results

We evaluate and compare the performance of SA-DSR to that of standard DSR in terms of five metrics as follows:

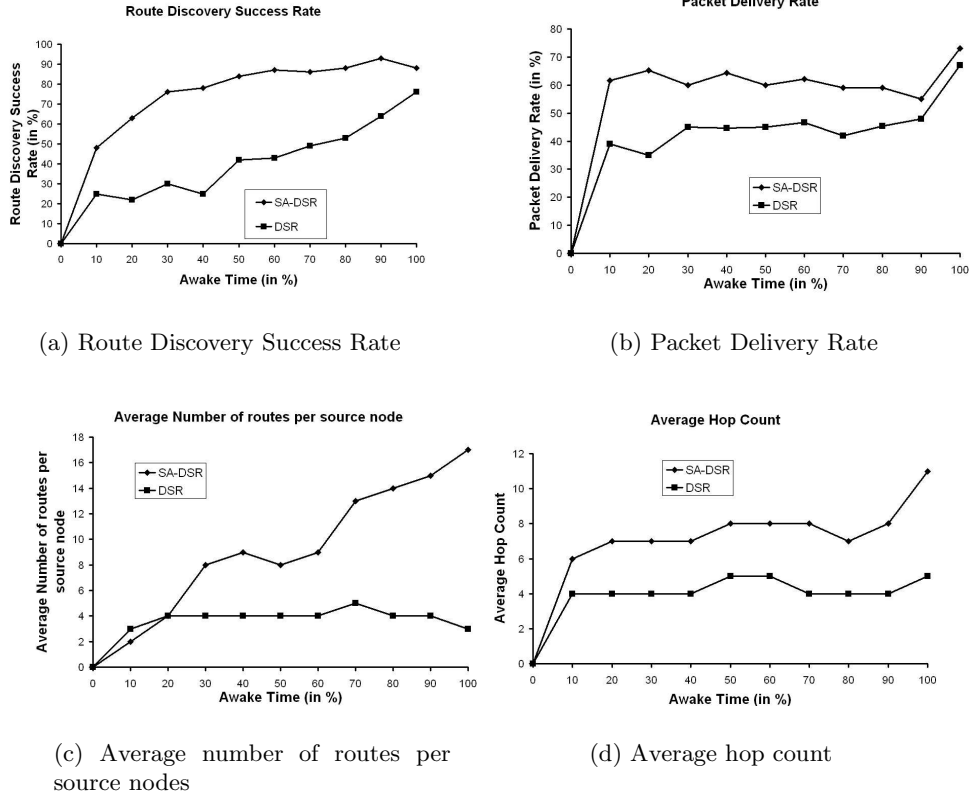


Fig. 1. simulation results for SA-DSR

In Figure 1(a), we measure *the route discovery success rate* for both of the protocols. We define route discovery success rate as the ratio of number of successful source nodes getting RREP packets back to the number of source nodes sending a RREQ packet. This graph gives an impressive performance improvement of SA-DSR over standard DSR (up to 60%). It is expected because SA-DSR uses both the min-hop and stability metrics in route discovery and finds more routes than standard DSR, which never takes into account the link down times of nodes along the source route. At awake percentage of 100%, DSR should have the same route discovery success rate, the slight difference between the DSR and SA-DSR performance is due to the difference in network connectivity during various simulation trials.

In Figure 1(b), we measure *the packet delivery rate* of both protocols. We define packet delivery rate as the ratio of number of data packets received at the sink node to the number of data packets transmitted by source nodes having a route in their routing table after a successful route discovery. In all of the cases SA-DSR provides better packet delivery rate than DSR (up to 31%). Even with lower awake percentage

such as 10% SA-DSR gives 20% better packet delivery ratio than standard DSR. Three noticeable patterns are revealed in this figure:

1. Figure 1(b) shows that for lower awake percentages standard DSR has nearly 40% of packet delivery rate, but Figure 1(a) shows that it has only 25% of route discovery success rate. Although these two measurements seem to contradict each other, this anomaly can be explained as follows. When an intermediate nodes on a particular source route forwards a RREP back in standard DSR, it updates its own routing table with the partial route from itself to the target node. Thus the intermediate nodes often get a route to the target node without having to initiate any route discovery. Since in our simulation scenario there is only one sink node, such occurrences are naturally expected. But since such nodes do not initiate any route discovery themselves, they are not considered in the route discovery success rate. But they can successfully send data packets to the target node, thus leading to some cases where packet delivery rate is higher than route discovery success rate. But SA-DSR still offers better performance than DSR instead of inflated packet delivery rate of the latter protocol.
2. Figure 1(b) shows that SA-DSR offers nearly 65% packet delivery rate, but Figure 1(a) shows that it offers nearly 85% of route discovery success rate. This drop occurs because the MRSP algorithm is not implemented at the RREP phase and stability predictions during the RREQ phase do not consider the length of the route, leading to some false route stability predictions. With MRSP implemented at the RREP phase, SA-DSR performance in Figure 1(b) should be like that in Figure 1(a).

In Figure 1(c), *the average number of routes per node* has been depicted. It shows how SA-DSR finds more routes per node (almost more than 3 times in some cases) with increasing awake percentage.

In Figure 1(d), *the average hop count* has been measured for both protocols. It shows quite expectedly that the hop count for SA-DSR is much higher than standard DSR since the former finds multiple possible alive routes instead of only the min-hop one as in the case of standard DSR.

6 Conclusion and Future Work

In this paper, we introduce Stability Aware DSR (SA-DSR), which introduces DPM awareness into the routing decisions and finds multiple stable routes to the target node. We compare it to standard DSR, which does not consider power-saving but optimizes routing for shortest delay. We show that SA-DSR provides a significant increase in successful packet transmissions with comparable route establishment and maintenance overheads.

Future work will consider the following cases:

- *Probabilistic Stability*: We understand that there might be some specific scenarios in which the RSP algorithm in an intermediate node finds all routes to the sink node to be unstable and hence discards all corresponding RREQs, so that the

source node does not get any stable path to the sink node in response to its route request. But in case of non-time-critical applications even a less aggressive approach could be acceptable, i.e., an intermediate node measures a probabilistic stability of the routes and forwards all the RREQs which gives stability guarantees above the desired level. The desired level could be input by the users.

- *Multi Constraint Routing*: SA-DSR finds multiple stable routes for a particular target node and picks the min-hop one from the set of stable routes. While SA-DSR combines the two metrics stability and hop count, our future work will study how the combination of multiple metrics such as real-time deadline-aware or residual-energy aware routing with stability aware routing influences the performance of the protocol, i.e. we plan to make the protocol a multi-constraint one.

References

1. J. Monteiro, A. Goldman, A. Ferreira, "Performance Evaluation of Dynamic Networks using an Evolving Graph Combinatorial Model", *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2006.
2. Kwan-Wu Chin, John Judge, Aidan Williams, Roger Kermode, "Implementation Experience with MANET Routing Protocols", *ACM SIGCOMM*, 2002.
3. C. K. Toh, "Maximum Battery Life Routing to support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks", *IEEE Communication Magazine*, June 2001.
4. M. Tarique, K. Tepe, M. Naserian, "Energy Saving Dynamic Source Routing for Ad Hoc Wireless Networks", *Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, 2005.
5. V. Marbukh, M. Subbarao, "Framework For Maximum Survivability Routing for a MANET", *MILCOM*, 2000.
6. V. Venugopal, R. Bartos, M. Carter, S. Mupparapu, "Improvement of Robustness for Ad Hoc Networks Through Energy Aware Routing", *Parallel and Distributed Computing and Systems*, 2003.
7. Morteza Maleki, Karthik Dantu, Massoud Pedram, "Lifetime Prediction Routing in Mobile Ad Hoc Networks", *Proceedings of IEEE Wireless Communications and Networking Conference*, 2003.
8. Carla F. Chiasserini, Ramesh R. Rao, "A Distributed Power Management Policy for Wireless Ad Hoc Networks", *Proceedings of IEEE Wireless Communication and Networking Conference*, 2000.
9. Suresh Singh, Mike Woo, C. S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks", *Proceedings of MobiCom 98 Conference*, Dallas, 1998.
10. David B. Johnson, David A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.