ARM: Anticipated Route Maintenance Scheme in Location-Aided Mobile Ad Hoc Networks

Seungjin Park, Seong-Moo Yoo, Mohammad Al-Shurman, Brian VanVoorst, and Chang-Hyun Jo

Abstract: **Mobile ad hoc networks (MANET) are composed of moving wireless hosts which, within range of each other, form wireless networks. For communication to occur between hosts that are not within each other's range, routes involving intermediate nodes must be established; however, since the hosts may be in motion, a host that was part of a route may move away from its upstream and downstream partners, thus breaking the route.**

In this paper, we propose anticipated route maintenance (ARM) protocol with two extensions to route discovery based routing scheme: Extend the route when nodes on a link move apart from each other and they have common neighbor that can be "inserted" in the path, and shrink route when a node discovers that one of its neighbor which is not the next hop is also on the same route several hops later on. By utilizing only local geographic information (now a part of some route finding algorithms), a host can anticipate its neighbor's departure and, if other hosts are available, choose a host to bridge the gap, keeping the path connected. We present a distributed algorithm that anticipates route failure and performs preventative route maintenance using location information to increase a route lifespan. The benefits are that this reduces the need to find new routes (which is very expensive) and prevents interruptions in service. As the density of nodes increases, the chance to successfully utilize our route maintenance approach increases, and so does the savings.

We have compared the performance of two protocols, pure dynamic source routing (DSR) protocol and DSR with ARM. The simulation results show how ARM improves the functionality of DSR by preventing the links in the route from breaking. Packets delivery ratio could be increased using ARM and achieved approximately 100% improvement. The simulations clarify also how ARM shows a noticeable improvement in dropped packets and links stability over DSR, even though there is more traffic and channel overhead in ARM.

Index Terms: **Location-aided, mobile ad hoc networks (MANET), reactive, route maintenance.**

I. INTRODUCTION

A network that consists of wireless mobile hosts without any centralized control point or fixed infrastructure is called a wireless ad hoc network [1], [2]. Since these hosts may move, the network is dynamic—causing the topology of the network to change in unpredictable ways. Each mobile host (or node) has a limited transmission range. Therefore, if a node (source node) needs to communicate with another node (destination node) that is not within its range, the source node must find the intermediate nodes (*bridge nodes*) that will relay the message all the way to destination (called a *route* or *path*). Finding and maintaining such routes in ad hoc networks are very challenging tasks due to mobility of the hosts. *Route failure* occurs when an intermediate node can no longer communicate with its upstream or downstream neighbor. When a route fails, time will be spent finding a new replacement route. The term *route lifespan* refers to the amount of time the route can function without failing.

A popular classification of the routing algorithms is based on the time when the route is determined. In *proactive* algorithms [3], each node constantly updates and maintains the routes to all nodes in the network. Another class of algorithms called *reactive* algorithms [4]–[6] start finding routes only when necessary. Recently developed routing algorithms [7]–[9], [23] are the examples that adopt approach using geographic information via global positioning system (GPS) or other means. Location-aided routing (LAR) [7] uses reactive approach when a node in the network wants to find the position of a destination, and distance effect algorithm for mobility (DREAM) [9] uses proactive approach by constantly exchanging position information among nodes of the network. Greedy perimeter stateless routing (GPSR) [8], [23] introduces a novel method called perimeter routing that guides the packet when there is no next available node in greedy forwarding. Even though both methods use global flooding to find the destination nodes for the first time, the position information of the nodes can reduce the considerable amount of search space for later search.

The route discovery process in most algorithms uses two types of control packets: Route request packet (REQ) and route reply packet (RPY). When a node s wants to send a message to another node d, s issues a REQ that is flooded through the ad hoc network. When d receives the REQ packet, it sends RPY packet back to s informing that the path has been established. On receiving the RPY, s starts to send the data packets.

Finding a route in wireless networks requires considerable resources (time, bandwidth, and power) because it relies on broadcasting. Therefore, it makes sense to protect this investment. However, most of the protocols described above do not put much effort into path maintenance. Ad hoc on demand distance vector (AODV) [6] and dynamic source routing (DSR) [4] retain only one path from any source and destination pair. When a link is broken, the upstream node of the link propagates the information to all source nodes. On receiving the notification of a broken link, source node can restart the route finding process

Manuscript received July 28, 2003; approved for publication by Dong-Ho Cho, Division III Editor, February 23, 2005.

S. Park is with the Department of Computer Science, Michigan Technological University, USA, email: spark@mtu.edu.

S.-M. Yoo and M. Al-Shurman are with the Electrical and Computer Engineering Department, University of Alabama in Huntsville, USA, email: {yoos, al-shum}@eng.uah.edu.

B. VanVoorst is with the Honeywell Labs. in Minneapolis, USA, email: brian.vanvoorst@honeywell.com.

C.-H. Jo is with the Department of Computer Science, California State University Fullerton, USA, email: jo@ecs.fullerton.edu.

Table 1. Location notation.

Formula	Description
$(x_p, y_p)_i$	position of node p at time t_i
$v_i(p)$	velocity of p at time t_i
$d_i(p)$	p's direction of movement at time t_i
$\|(p,q)\|_i$	distance between p and q at time t_i

if the route is still desirable. Temporally ordered routing algorithm (TORA) [2] takes a different approach and maintains all the paths from source to destination pair and uses the most promising path for packet delivery. When a link in the path is broken, alternative paths to the destination will be used with possible reconfiguration of links according to the height values associated with each node. The protocols described above start to search for a new path only after some link is broken, which may cause unexpected delay in packet delivery. This is not desirable especially for the systems that support QoS. Route maintenance problem has keen attention in recent literature [14]–[17].

Contrary to those proposals, we want to use the local information only based on anticipation of the failure of routes. In this paper, we propose anticipated route maintenance protocol (ARM) that will allow routes to have longer route lifespans without interrupting service by replacing as unsafe link with stable one(s). The approach is best integrated with reactive routing algorithms for hosts that have geographic location information. In ARM, a predefined value T (expected time to find replacement link(s) for a unsafe link) is very important. We have simulated the performance of two protocols, pure DSR and DSR with ARM, and the simulation results show how ARM improves the functionality of DSR by preventing the links in the route from breaking. Packet delivery ratio could be increased using ARM maintenance and achieve approximately 100% improvement. The simulations clarify how ARM shows a noticeable improvement in dropped packets and links stability over DSR, even though there is more traffic and channel overhead in ARM. The overhead could be controlled by choosing the right value for T which depends on the node density in the network. After all, we have found that ARM is an efficient scheme in wireless ad hoc networks because of its preventive action to save the links before we lose any packet.

This paper is organized as follows. Section II introduces background (notation) of the ARM scheme. Section III introduces the ARM scheme in details. Section IV explains the simulation environments. Section V shows the detailed simulation results. Finally, Section VI makes a concluding remark and mentions further research.

II. BACKGROUND

A. Notation

Quite often, a node (source node) s wants to send a packet to another node (destination node) d that is not a neighbor of s. For a successful transmission there must be a series of *bridge nodes*, $b_1, b_2, \cdots, b_m$, that relay the packet from s to d. These ordered set of nodes $(s = b_0, b_1, b_2, \cdots, b_m, d = b_{m+1})$ is referred to as a *path* from s to d and denoted as $P(s,d)$. A link connecting

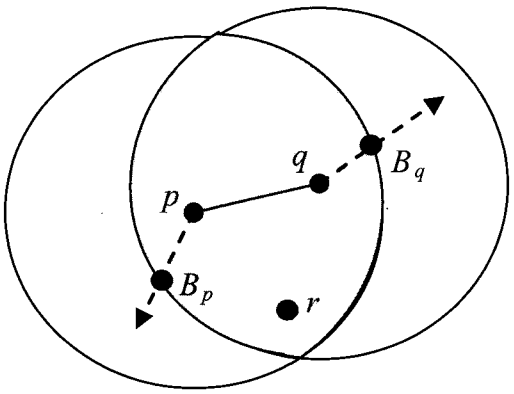


Fig. 1. Node p and q are in range of each other, but their movement will cause their link to break. Node r can bridge the gap. B_p and B_q are the locations p and q will be at when the link breaks.

two consecutive bridge nodes b_i, b_{i+1}, is denoted as (b_i, b_{i+1}). We use $p > q$ to indicate that node q appears later than node p in the path. For any two neighbor nodes, b_i and b_{i+1}, in a path, b_i is called *upstream* of b_{i+1}, and b_{i+1} is *downstream* of b_i. Note that every bridge node, b_i, has exactly two neighbor nodes, b_{i-1}, b_{i+1}, in the path. We assume that all nodes are in 2D plane. Some of the values related to nodes in the location-aided mobile networks are shown in Table 1.

Note that the speed and direction of a node can be deduced from the position of the node at time t_i and t_{i+1}. Nodes p and q are said *diverging* at time i, if $|(p,q)|_i < |(p,q)|_{i+1}$, and *converging* otherwise. Refer to Fig. 1. Let R_p and A_p denote the radius and the area of the transmission range of node p, respectively. Link $(p,q)_i$ is said to be in an *unsafe state* if it is predicted to break in time less than or equal to T (the value of T is described in Section III). We predict a link (p,q) to break if it meets two conditions: 1) $|(p,q)|_{i+T} \geq R$, and 2) $d_i(p)$ and $d_i(q)$ are diverging. (The expected time that the link will break is the same for both p and q.)

B. Previous Work

Lee and Gerla [18] proposed AODV-BR algorithm. This algorithm utilizes a mesh structure to provide multiple alternate paths to existing on demand routing protocols without producing additional control messages. Data packets are delivered through the primary route unless there is a route disconnection. When a node detects a link break, data packets can be delivered through one or more alternate routes and are not dropped when route breaks occur. Route maintenance is executed utilizing alternate paths. In this algorithm, the alternative route is utilized only when a primary link is broken while, in our proposed scheme, the route is dynamically established before a link is broken. Therefore, message delay will be less in our scheme. Further, their scheme may not work if the alternative route fails as well.

Li and Mohapatra [19] introduced LAKER, a location-aided knowledge extraction routing. This approach reduces the flooding overhead in route discovery by extracting knowledge of the nodal density distribution of the network and remembering the series of locations along the route where there are many nodes around. However, this scheme does not deal with route maintenance.

Stojmenovic *et al.* [20] proposed depth first search and location based localized routing and QoS routing in wireless networks. This scheme considers a connection time (estimated lifetime of a link) as a QoS criterion. Here, the method for estimating time of disconnection was mentioned, and the method is similar to ours.

Stoimenovic [21] proposed location update scheme for efficient routing in ad hoc wireless networks. Here, location updates are done when a certain pre-specified number of links incident to a node have been established or broken since the last update. The paper claims that distance based updates and movement based updates may have limited usefulness in ad hoc networks. It suggests an update when a certain pre-specified number of links incident to a node have been established or broken since the last update. In our proposed scheme, an expected time to replace link(s) for a unsafe link is used instead of the number of links established/broken.

Stoimenovic [22] reviewed many position based routings in ad hoc networks. The accuracy of the destination's position is an important problem to consider for efficient routing. It is shown that routing protocols that do not use geographic location in the routing decision may not be scalable. It is likely that only position based approaches provide satisfactory performance for large networks. Greedy mode routing was shown to nearly guarantee delivery for dense graphs, but to fail frequently for sparse graphs since the destination is also moving and it is not clear where to send message. The routing process is converted from the greedy mode to recovery mode at a node where greedy mode fails to advance a message toward the destination. Our proposed scheme is scalable since the scheme is based on distributed, local geographic information.

III. ARM: ANTICIPATED ROUTE MAINTENANCE SCHEME

In this section, we present an algorithm that efficiently maintains the paths that are already established, despite the movement of the nodes. The assumptions we make are:

1) A reactive routing algorithm exists. The LAR can be a candidate because it uses geographic information and reduces the need for flooding. More candidates based on Voronoi diagram and convex hull are shown in [24].
2) Each node knows its current position, speed, and direction of moving (i.e., with GPS).
3) Each node maintains a table that contains the positions, speeds, and directions of movement of all its neighbors.
4) Each node also maintains a *route table* that contains all routes passing through it. The content of the table includes the addresses of the route source, destination, and upstream and downstream nodes for each route.
5) Each node in a route knows its order in the route based on hop count.
6) All links are bidirectional and all nodes have the same transmission range.
7) The links of candidate nodes for expand and shrink routines (explained later) are not broken during the execution of the routines. Also, the batteries of the candidate nodes are not drained out during the execution of the routines.

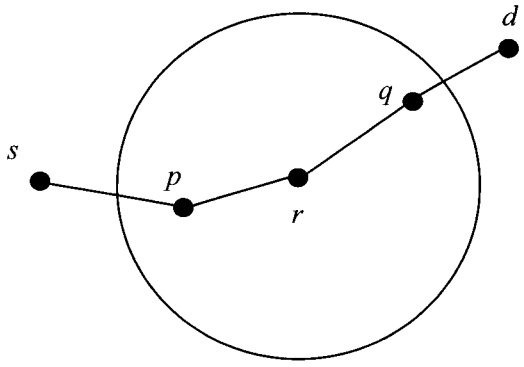


Fig. 2. Expand algorithm finds node r to fill gap between p and q, allowing path to stay alive. Node r's communication range is represented by the circle.

Not all of these assumptions are strictly necessary, for example, 6) above can be relaxed if every pair of neighbor nodes select only bidirectional links between them by handshaking.

Based on local geographical information stored at each node, the proposed algorithm, ARM, predicts any route failure due to node movement, and therefore, is able to perform dynamic route maintenance. ARM consists of two routines: Expand and shrink. We explain each routine in detail below.

A. Expand Routine

Since nodes in ad hoc networks are migratory, if two adjacent nodes in a route move out of each other's transmission range, then the link is broken, and so is the route. Let p and q be upstream and downstream nodes of an unsafe link, respectively. The main function of the expand algorithm is to prevent the route from breaking (thus saving the network from having to undergo an expensive search for a new route). To do this, we must find additional bridge nodes between p and q before the break occurs and adjust the route accordingly. Refer to Fig. 1.

Suppose link connecting node p and node q, $p > q$, becomes unsafe at time t_u and is estimated to break at time t_b. Since each node has a table containing information about all neighbor nodes, node p selects a bridge node r from $A_p \cap A_q$ at time t_b, which is shown in Fig. 2. (This region can be calculated based on the geographic information of p and q.) Then, just before the disconnection of the link, node p informs node r that r has been chosen as part of the route. Note that since q is still in the range of p, it can listen to the notification from p to r, and prepares to receive packets from r. Therefore, after q moves to position q', the new route will be $(s, \cdots, p, r, q \text{ at } q', \cdots, d)$.

In general to achieve long lifespan of paths and prompt delivery of packets, a bridge node, b, for (p, q) should be chosen such that the two resulting links (p, b) and (b, q) last longer than the results of other choices. Therefore, for all nodes $b_1, b_2, \cdots, b_m \in A_p \cap A_q$, the chosen bridge node should satisfy $\max(\min(\text{lifespan of } (p, b_i), \text{lifespan of}(b_i, q)))$, for all $b_i \in \{b_1, b_2, \cdots, b_m\}$. (Lifespan can be calculated using formula in Section III-A.2.) If the chosen node is "busy" by being part of many paths already, the node forming the links with the next longest lifespan can be selected. Further enhancement on this topic will be discussed in Section III-C.

If there is no bridge node available in $A_p \cap A_q$, then there are two possibilities to save the route: 1) Find a route from p to q when q will be at location $(x_q, y_q)_b$ using multiple nodes or 2) find a route from p to q's downstream neighbor that does not use q at all (a partial new path). We devote our research to Case 1, as our simulation shows that this has more payoff. Case 2 can be performed by simply using the existing routing algorithm and will not be discussed here.

To do Case 1, we must select a bridge node that is outside of $A_p \cap A_q$ at time t_b because no node is available in this intersection. We are looking for two bridge nodes, b_1 and b_2, that will form a path to q at $(x_q, y_q)_b$. Node b_1 must be within the range of p now, so p will broadcast a control message to all nodes in it's range requesting a reply if they have a neighbor b_2 that will be in position to talk to q at $(x_q, y_q)_b$. If there are nodes b_1 and b_2 that fulfill these requirements, we can form a bridge path p, b_1, b_2, that will keep the link alive. Node p then notifies b_1, which in turn notifies b_2, of the new link. If p is unsuccessful in fixing the anticipated failure, p has the option of notifying another node j, where $j > p$, of the failure and allowing it to execute the expand algorithm.

A.1 Deciding When to Invoke Expand

A bridge node has to be chosen before the link is broken in order to keep the packet flow from being interrupted. An important question to answer is when the bridge node is chosen and starts to receive/send packets to next node in the path. Recall that link $(p, q)_i$ is said to be in an *unsafe state* if it is predicted to break in time less than or equal to T. T is the estimated time it would take to find the necessary bridge nodes and depends on not only the density of the network around p and q but the speeds of p, q, and all their neighboring nodes. T is an experimental value from an average time to fail (TTF) from all the nodes. As the network becomes sparse, the value of T gets larger. This is because if network is sparse, and therefore no node is available to be a bridge node for (p, q), then p has to inform its upstream node in the path to find a new path to q or q's descendents. Since this process may take longer time than p's finding a bridge node, T should be larger. In Section V, we will report our simulation results based on various values of T inversely proportional to the number of nodes in p's range. Note that whenever a link becomes unsafe, the expand routine is invoked and executed, regardless of whether the path is currently being used for communication or not.

A.2 Determining the Position When a Node Becomes Unsafe

As shown in Fig. 1, suppose current positions of two neighbor nodes p and q at time t_i are $p_i = (x_{pi}, y_{pi})$ and $q_i = (x_{qi}, y_{qi})$, respectively. B_q (B_p, resp.) is the intersection point of d_q's (d_p's, resp.) trajectory and boundary of $A_p \cap A_q$, i.e., where node q will leave the range of p and break the link. The positions of p and q at time t_j, where $j > i$, become $p_j = (x_{pj}, y_{pj})$ and $q_j = (x_{qj}, y_{qj})$, where $x_{pj} = x_{pi} + (t_j - t_i)v_p \cos\theta$, $y_{pj} = y_{pi} + (t_j - t_i)v_p \sin\theta$, $x_{qj} = x_{qi} + (t_j - t_i)v_q \cos\alpha$, and $y_{qj} = y_{qi} + (t_j - t_i)v_q \sin\alpha$. (θ and α are angles of d_p and d_q, respectively, measured in polar coordinate system.) Let R be the transmission range. Therefore, the time for node p (q, resp.) to reach B_p (B_q, resp.) from current position is T_{pB} (T_{qB}, resp.) = $SQRT\ ((R^2 - (x_{pi} - x_{qi})(x_{pi} - x_{qi} + 2) - (y_{pi} - y_{qi})(y_{pi} - y_{qi} + 2))/(v_p \cos\theta - v_q \cos\alpha)^2 + (v_p \sin\theta - v_q \sin\alpha)^2)$, i.e., the amount of time left until the break will occur. Therefore, with the given value of T_q (recall that T_q is the predefined amount of the time taken for link $(p, q)_i$ from *unsafe state* to broken state), the position of node $q = (x_q, y_q)$, where q falls into an unsafe state, is $x_q = x_{qi} + (T_{qB} - T_q)v_p \cos\alpha$, $y_q = y_{qi} + (T_{qB} - T_q)v_p \sin\alpha$.

A.3 Managing Path Length

It is possible that the expand routine can lead to very long, but useable, paths. We propose a simple heuristic by which a source node may choose to find another path. Suppose source node, s, knows the number of hops, H, on it's path to destination, d. (To find H, s may need to send a control message to d, and have it returned.) Let R_{AVG} be the average transmission ranges of nodes in the network. If $H \times R_{AVG} > r \times |(s, d)|$, where r is a predefined value to prevent the paths from becoming unnecessarily long, it is better to abort the path and rebuild it. Our simulation results show that the proper r value lies between 1.5 and 2, and in our experiments traffic reduces as much as 17%.

B. Shrink Routine

The shrink routine handles the case where bridge nodes in a long path have come close to each other, allowing for a possible shortcut. This optimization is useful in general, but also serves to help "prune" paths that may become long due to the expand phase.

The shrink algorithm requires that each node record its position (in terms of hops) for each route it is a part of. That is to say that a route with n nodes, has hop-positions $0, 1, 2, \cdots, n-1$. If a node n_j is in position j, of the route it stores value $j \times 1000$ as it's position in the route. (The reason for the multiplication will be explained later.) We call this value the *position in the routing order*.

In order for the shrink phase to work, nodes must exchange some route table information when they come within range of each other. Specifically, when two nodes n_i and n_j come within range of each other, they will exchange control packets that contain the routes they are a part of, and their position in the routing order. If nodes discover that they are both part of the same path then they are called *friend nodes*.

There are two cases for shrinking the path. The easiest (and probably most common) is if two (and only two) nodes discover that they are friend nodes at the same time. If n_i and n_j are friend nodes they can shorten the path by directly routing from one another (skipping $n_{i+1}, n_{i+2}, \cdots, n_{j-1}$). A more pathological case exists when several nodes determine they are all friend nodes at the same time. If there are f friend nodes, we could pick $(f^2 - f)/2$ different shortcuts. However by picking the node *earliest* in the path and making a route to the node *latest* in the path (using the position in routing order) will be the *best* shortcut. For example, if nodes p, q, and r with route positions 3000, 8000, and 11000, respectively, all discover that they are friend nodes at the same time, and node p should route

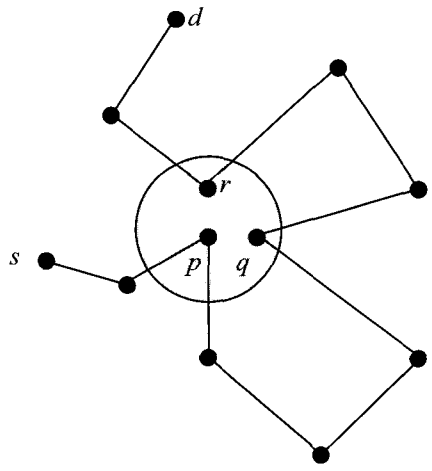
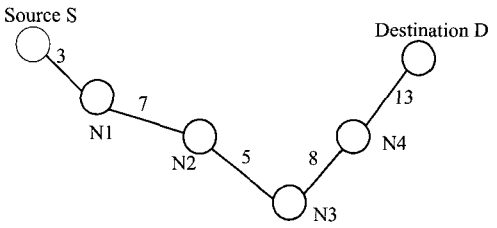


Fig. 3. Shrink case with three friend nodes. Node p will shortcut to r. This will reduce the path by six hops.



T value (sec)	Hop S-N1	Hop N1-N2	Hop N2-N3	Hop N3-N4	Hop N4-D
1	☐	☐	☐	☐	☐
3	☑	☐	☐	☐	☐
5	☑	☐	☑	☐	☐
7	☑	☑	☑	☐	☐
9	☑	☑	☑	☑	☐
11	☑	☑	☑	☑	☐
13	☑	☑	☑	☑	☑
15	☑	☑	☑	☑	☑
20	☑	☑	☑	☑	☑

☐ ARM expand will not satisfy the conditions so it will not do the maintenance.
☑ ARM will run.

Fig. 4. An example of various T values and their effects on route maintenance.

directly to node r to skip the most intermediate bridge nodes. This is shown in Fig. 3.

In order to allow for intermediate nodes that were not a part of the original route (i.e., due to the expand algorithm), we must assign to them a position in the routing order. By multiplying the hop count by 1000, we allow considerable resolution for this to take place. Consider two nodes that are a part of a path, p and s, which have positions in the routing order of 4000 and 5000 respectfully. Imagine p and s are moving such that they need a bridge node between them. Two nodes q and r are found that can fill this role (resulting in the path p, q, r, s). We must give q and r positions in the route order so that their relative position can be determined. We can assign q to have position 4333 and r to have 4666. If later on a bridge node is needed between p and q, q and r, or r and s, there is sufficient room to select one.

C. Communication Requirements for ARM

ARM does not require much additional overhead to accomplish its task. During expand, a node must only communicate to its upstream and downstream neighbor its velocity and direction of movement. After initially communicating this information, it need not communicate this information again unless it changes direction or velocity significantly. By limiting these exchanges to only required updates, this communication happens infrequently. If a node is in an unsafe state, it will need to locate another node within its broadcast range to bridge the potential break. This communication is with nearest neighbors and is only done when needed. Using this approach, ARM does not require scheduled periodic communication.

The communication for the shrink routine only happens when two or more nodes come within range of each other, when previously they had been separated. This communication is only among nearest neighbors, and strictly speaking is only necessary to take advantage of the shrink algorithm benefits. Expand and shrink can operate independently of each other.

D. ARM and T Value

The ARM protocol runs periodically in every node. First, it calculates the expected time that the link connecting its downstream node would break. Then, if this time is less than or equal to predefined value T (expected time to recover from failure in worst case), the ARM-expand will perform the expand routine and try to find a bridge node before a failure occurs and consequently without any interruption in service. Fig. 4 shows an example of a simple path in an ad hoc network. Here, a number corresponding to a link is the expected TTF for the link. The effect of changing the value of T on the running of ARM-expand is shown in the figure. Suppose that we choose a large value for T, the network is assumed to be sparse. We need a large time to find a new bridge, so ARM-expand will be active most of the time and maintaining the routes from failure. When we choose $T = 20$, TTFs of all hops will be less than T and ARM-expand will perform maintenance and run for each hop. If we choose a small value of T (for example, $T = 1$), we can see that no TTF will be less than T, so ARM-expand will not run this time. Note that the actual time to recover the route is not known and difficult to measure. Therefore, we can use ARM-expand to measure the estimated time to recover the route.

The question arises why we do not assign T a large value and make ARM always perform the maintenance. If we choose a large value for T, ARM will expand routes that are stable and increase the path length (add redundant nodes). In addition to performing maintenance, the ARM needs to send control packets, which will increase the channel overhead. On the other hand, if we choose a very small value for T, ARM may not perform the expand routine in most cases, and therefore no performance improvement due to failure avoidance is expected. Thus, one motivation of simulating ARM is to find a proper value of T, and consequently to show how ARM can improve any routing algorithm, especially DSR.

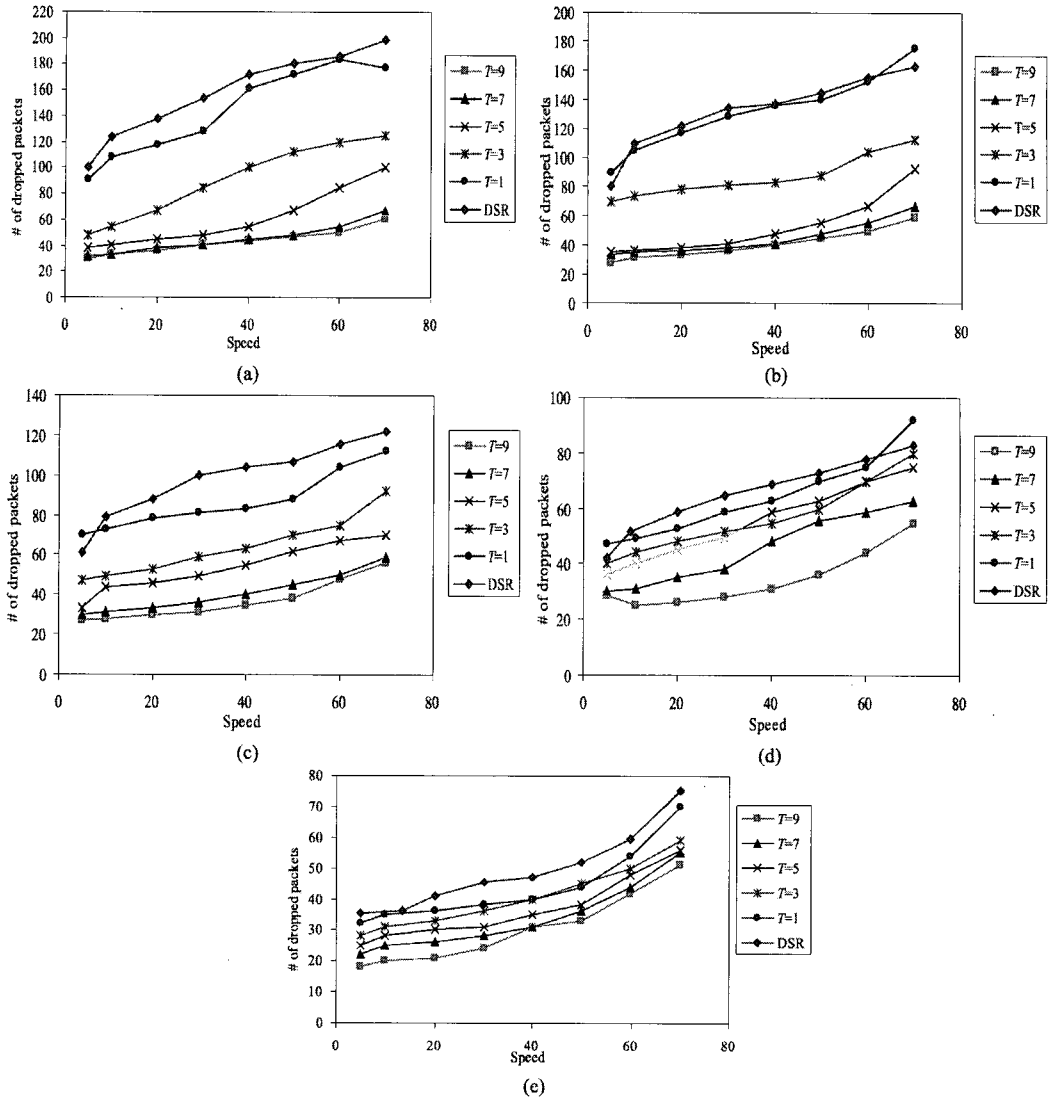


Fig. 5. Dropped packets for ARM and DSR with increasing number of nodes (N) with increasing expected recovery times (T): (a) $N = 50$, (b) $N = 40$, (c) $N = 30$, (d) $N = 20$, (e) $N = 10$.

IV. SIMULATION ENVIRONMENT

In our simulation model, ns-2 is used [10]. It is a discrete event simulator developed at UC Berkeley where networking research is targeted. ns-2 provides substantial support for simulation of TCP, routing, and multicast protocols, such as UDP, TCP, RTP, and SRM over wired and wireless (local and satellite) networks.

To compare the performance of pure DSR (will be referred to as just DSR) with ARM on top of pure DSR (will be referred to as just ARM unless otherwise specified), a hypothetical network is constructed for the simulation purpose and then monitored for different number of parameters, such as the number of mobile nodes, movement speed, and expected time to recover T. We simulate our model for different number of mobile nodes, from 10 to 50, with 10 nodes incremental. Speed is also varied at the range of 5 to 70 km/h. Each mobile node in the MANET is assigned an initial position within the simulation area of 670 $\times$ 670 meters. The simulation takes place for 900 second for every run. Nodes are normally distributed when initialized and the initial position for the node is specified in a movement scenario file created for the simulation using a feature within the ns-2. The nodes randomly move within the simulation area. During the simulation, every node is in a continuous movement (i.e., zero pause time) to resemble the real highly dynamic network and to evaluate the ARM.

During the simulation, each node can send and receive packets at a random pattern, that is, a node can send a packet at any specified time scheduled by the simulator but can not receive while it is in a sending mode. Each node is equipped with a single transceiver radio antenna that can either send or receive packets at a time but not both. With a bandwidth of 2 Mbps, this has no lagging effects to our simulation results because the node traffic generator used here is a typical *constant bit rate* (CBR). It is programmed to generate four packets every second with maximum packet size is 4 kB. Thus, during only 64 millisecond per second, the node transceiver can be busy sending its own packets. Most of the time the node will be listening to the media to

receive and forward packets for the network.

The radio transmission range for a node covers a circle of a radius of 50 meter, and the MAC layer uses IEEE 802.11 protocol with *carrier sense multiple access with collision avoidance* (CSMA/CA) as shared media access protocol to avoid the hidden terminal problem [11]. When a node wants to send a packet, it sends first a short *ready-to-send* (RTS) packet to the receiving node, the receiving node in turn will reply with a *clear-to-send* (CTS) packet to inform the sending node it is ready, and then the transmission takes place. When the transmission successfully takes place, the receiver sends an acknowledgement (ACK) to inform the sender upon reception [12]. The simulation is event-driven, and all the simulation events will be logged in a trace file and then processed using *tracegraph* software [13] to extract the needed statistics.

In this simulation, we will study some performance metrics to compare the ARM with the standard DSR. The following are the main metrics to be used:

- Number of dropped packets which is an indirect measure to the number of disconnections occurred between the nodes during the simulation.
- Number of dropping nodes which measures the actual number of the nodes that show a disconnection during the data transmission.
- Average path length in hops to measure the effect of ARM on the path length and the packet delay.

There are other metrics we will consider in addition to those mentioned above, but we will give a clear explanation later when we use them.

V. SIMULATION RESULTS

A. Number of Dropped Packets

In the first part of the simulation, the number of dropped packets for both DSR and ARM is shown in Fig. 5 with varying node speed for different number of mobile nodes and variation of expected time to recover T. In all figures in this section, five different T values (1, 3, 5, 7, and 9) are simulated for ARM.

It is clear from the figure how the various node speeds can affect the number of dropped packet for both DSR and ARM. As the speed increases, the number of dropped packets will increase, but ARM shows an improvement by changing the value of T (the expected time to find new bridge node in worst case). As we increase the T value, the number of dropped packet decreases. The result holds regardless of N, the number of nodes. This is logical because the expand routine in ARM will fix the route by inserting bridge node, if the link connecting downstream node is detected to break within a period of time smaller or equal to T. Thus, if we assign a large value for T, the routine will run most of the time and expand the routes by inserting bridge nodes because all the links will satisfy the condition (TTF $< T$). On the other hand, the small value of T will prevent the expand routine from running, i.e., TTF $> T$. In this case, real maintenance protocol will be the same as DSR; this will lead curves for ARM to come close to DSR because our base protocol for ARM is DSR.

The shrink routine in ARM has no effect on the number of dropped packets in the network, because its only function is to reduce the path length by pruning any unnecessary nodes and shorten the path length. If ARM-shrink makes any route unsafe, ARM-expand will fix this and convert it to safe.

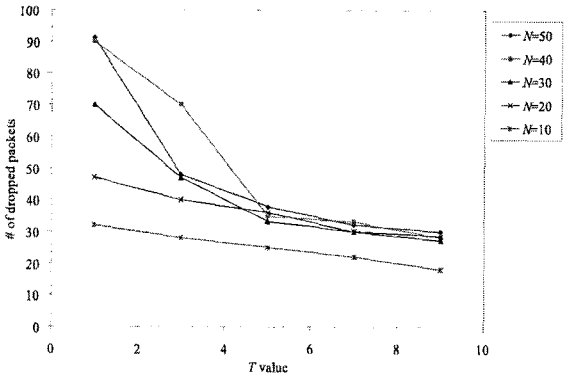


Fig. 6. Dropped packets for ARM and DSR with various N and T.

Another interesting observation from the simulation result is how the number of nodes in the network affects the choice for best value of T. Refer to Fig. 6. The smaller is the number of mobile node, the larger is the time to find new route because the network becomes sparse and no bridge node will be available to prevent the route failure. For example with $N = 20$, we can see that the number of dropping packets after $T = 7$ does not show much improvements. Here, the actual value of T, the recovery time for the link break is slightly less than 7 sec. Actually, there is difficulty to measure the exact value due to many factors such as the uncertainty of node direction, position accuracy from GPS system, and transmission fading loss. For another example with $N = 40$, we can see that the continuous, sharp decrease in the curve until the value of T is 5. After 5, the curve shows stability. The value of this "border" point relies upon the actual value for time to recover, to be more specific, it is slightly greater than the actual value. In our simulation, results show that the value of T is between 3 and 5 when $N = 40$, between 5 and 7 when $N = 30$, and between 1 and 3 when $N = 50$.

B. Number of Packet-Dropping Nodes

In this subsection, we will study the number of the packet-dropping nodes relative to mobile node speed and the variable T. As we mentioned earlier, the value of T is correlated with the network topology and nodes density. Refer to Fig. 7.

In the figure, it can be observed that the larger the number of nodes is, the smaller the ratio of packet-dropping nodes is for both protocols. For example, when $N = 10$ at speed 70 km/h and $T = 1$, we can see that nine out of ten are dropping nodes, that is, 90% of the nodes has dropped at least one packet, while ten out of ten (100%) are packet dropping nodes with DSR. By changing N to 50 nodes with the same configuration as above ($T = 1$), both protocols show an improvement in packet-dropping nodes number. Here, ARM shows 34 out of 50 (68%), while DSR shows 33 out of 50 (66%), nodes are packet dropping. With $N = 50$ and $T = 9$ with the same speed, ARM shows only 11 out of 50 (22%) nodes are packet-dropping. Thus, with the same configuration, ARM can reduce the packet-

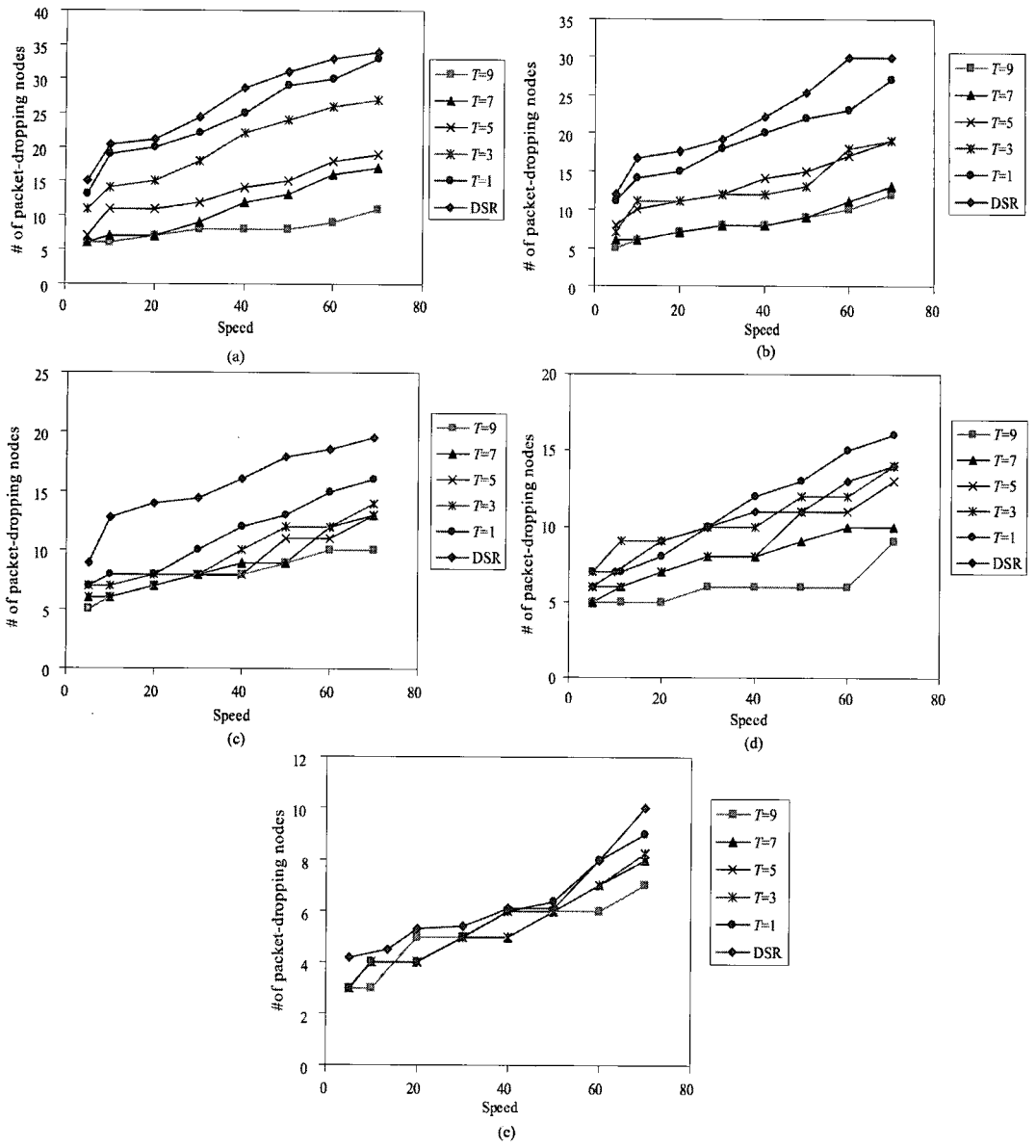


Fig. 7. Number of packet dropping nodes for ARM and DSR: (a) $N = 50$, (b) $N = 40$, (c) $N = 30$, (d) $N = 20$, (e) $N = 10$.

dropping nodes to one third relative to DSR if we choose larger value for T. Also, we can see from the plots above that the difference between curves becomes smaller as the number of the nodes in the network decreases i.e., the distance between DSR curve and ARM curves becomes smaller and sometimes the curves intersect.

C. Average Path Length

Next, we will study the effect of the ARM maintenance protocol on the average path length in hops between sender and receiver nodes since the longer path will lead to longer packet delay. As discussed before, the ARM-expand routine may increase the lengths of paths between the sender and receiver by adding more nodes to the paths (more hops), while the ARM-shrink routine tries to shorten the route by trimming the redundant hops.

Refer to Fig. 8. We can see, by increasing the number of nodes, the path length will increase gradually. It is reasonable, as the number of nodes increase, the ARM expand routine may find more available nodes within the transmission range of the upstream node to insert in the link with its downstream node. Thus, more nodes are expected to be pushed in the route to avoid the service failure. On the contrary, with a small number of nodes in the network, the upper stream node may not find any available node within its transmission range to fix the route. Thus the path length will not be affected, but this leads to higher probability of link failure. Another result we can see is that, by increasing the speed of the mobile node, the average path length will increase. Since the probability for time to recover (T) to exceed TTF for the links is high (high mobility makes TTF be small), the ARM-expand will be in duty to push bridge nodes continuously.

As the T value increases, many bridge nodes will be inserted into the paths by ARM-expand routine, which will cause the

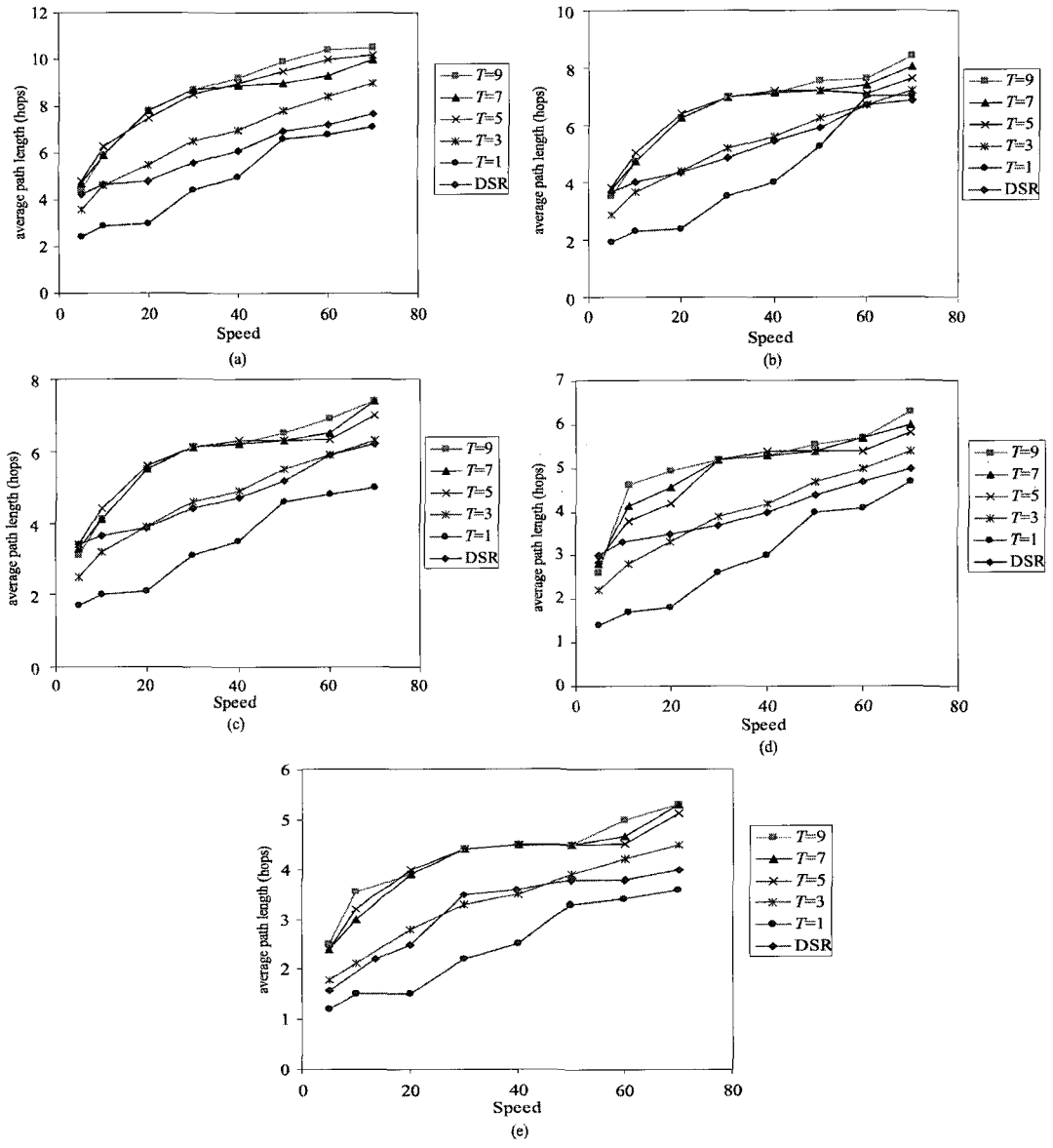


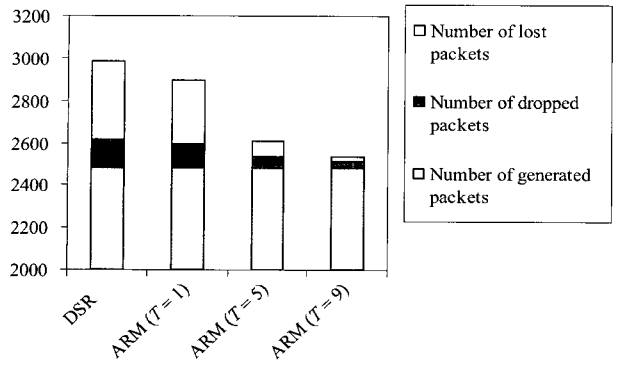
Fig. 8. Average path length for ARM and DSR for various values of T: (a) $N = 50$, (b) $N = 40$, (c) $N = 30$, (d) $N = 20$, (e) $N = 10$.

ARM curve to exceed the DSR curve. In our simulation, DSR curve resides between the ARM curves for $T = 1$ and $T = 3$ in most of the plots. With $T = 1$, average path length in ARM protocol is less than that of DSR, whereas it is opposite with $T = 3$ or above.

Now let us have a look on the effect of ARM-shrink on the path length. Since the path may become quite long due to ARM-expand routine, ARM-shrink will shorten the path by pruning the unnecessary links in the path.

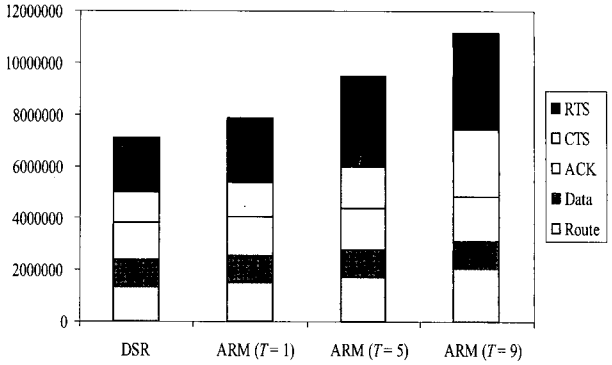
When T is chosen to be large, ARM-expand will be on duty, and regardless of the value of T, the ARM-shrink will be also on duty. With that large value of T, ARM-expand may increase the path length continuously and this leads to longer path length, whereas the ARM-shrink will prune the path and eliminate any unnecessary link, if found. However, ARM-shrink will not eliminate any bridge nodes inserted by ARM-expand if it causes any interruption in service to achieve QoS. The number of nodes added to the path is usually greater than the number of nodes eliminated by ARM-shrink.

When T is chosen to be very small it will lead the ARM-expand not to satisfy the condition (TTF $< T$), and the path length will not be affected by ARM-expand. However, ARM-shrink is always on duty, so the path will be shortened by ARM-shrink, whenever applicable. This is the reason why ARM curves for path lengths are distributed below and above the DSR curve. We can see that when $T = 1$, ARM curve resides below the DSR regardless of number of mobile nodes, and as the value of T increases, the ARM curves move up and exceed the DSR curve. With $T = 9$, the average path length in ARM is around the twice of the DSR path length.



	DSR	ARM ($T = 1$)	ARM ($T = 5$)	ARM ($T = 9$)
Number of generated packets	2481	2481	2481	2481
Number of dropped packets	137	118	58	36
Number of lost packets	367	297	72	20
Packet delivery ratio (%)	79.7	83.3	94.8	97.7

Fig. 9. Application layer statistics for various values of T.



	DSR	ARM ($T = 1$)	ARM ($T = 5$)	ARM ($T = 9$)
Route	1312030	1480230	1742030	2082030
Data	1086032	1077320	1056000	1026032
ACK	1420000	1500000	1580000	1750000
CTS	1175422	1355422	1625422	2595422
RTS	2140000	2450000	3520000	3710000

Fig. 10. MAC layer statistics for DSR and ARM.

D. Application Layer Statistics

In this subsection, we will show the application statistics for the same ad hoc network. Refer to Fig. 9. Here, the number of nodes in the network is 50 with the average speed of 20 km/h.

The figure shows that how efficient ARM is in terms of the number of both dropped and lost packets. Lost packets are different from dropped packets and defined as packets sent by the source and never received by the destination. With the same number of generated packets, we can see that the DSR can achieve a packet delivery ratio of 79.7% while ARM shows a better packet delivery ratio that ranges from 83.3% at $T = 1$ to 97.7% at $T = 9$.

We have noticed how the large value of T can increase the packet delivery ratio, but that does not mean that choosing a very large value for T is always good. After a certain value of T, the improvement in network performance is negligible. In addition, a large value of T means more traffic and more overhead in the network as we will explain next.

The overhead for ARM will be clear if we turn to study the MAC layer statistics. In order to expand the route we need to send a packet to the neighbor nodes and receive the reply from the nodes that received the packets, and then inform the downstream with the change. In addition, we need to send an acknowledgment for the bridge node. All theses packets produce more traffic and increase the overhead on the shared medium. Shrinking the routes needs the nodes to exchange some routing table information when they close within the range of each other, and this information exchange will increase the channel overhead. Next, we will compare the data on MAC layer.

E. MAC Layer Statistics

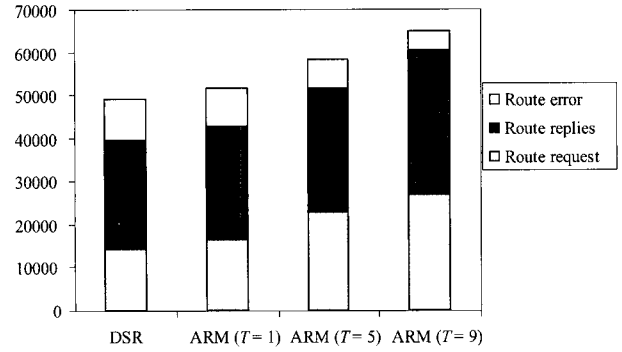
Fig. 10 shows the amount of MAC layer packets for both DSR and ARM with different values of T. It is based on 50 mobile nodes with a fixed speed of 20 km/h. Here, the number of MAC packets in ARM is much greater than that in DSR. Specifically, for routing packets the ARM will send more route request than DSR due to the longer paths, and so will the reply packets by the same reason. RTS/CTS/ACK are exchanged between sender and receiver nodes for reliable delivery of the packets. In ARM, these packets are more than those in DSR. For example, when $T = 9$, the RTS in ARM is almost twice compared to DSR. The data packets in MAC layer in DSR are more than those in ARM. As explained earlier, DSR shows more frequent link failure; thus, there are more data packets dropped and lost in DSR, and more data retransmission is required in DSR. Next, we will compare the data on network layer.

F. Network Layer Statistics

Refer to Fig. 11. The number of overall packets in ARM is greater than that in DSR. Note that the number of route error packets in ARM is less than that in DSR due to more packet losses in DSR that need to be fixed and notify other nodes about the faulty connection. On the other hand, the total number of both route request packets and route reply packets in ARM is larger than that in DSR, although the ARM sends request to find a bridge only to the node neighbors who can be reached in one hop request. In this case, every neighbor will reply to this request. Therefore, the total number of routing packets in ARM is larger than that in DSR.

G. Number of Breaks

In this and next subsection, we varied the number of nodes and transmission range (50 or 75 units). We assume that path lifespans range 2–10 timesteps that is relatively short. Therefore, efficient route maintenance is critical. Fig. 12 shows the



	DSR	ARM ($T = 1$)	ARM ($T = 5$)	ARM ($T = 9$)
Route request	14358	16485	22854	26985
Route replies	25253	26390	28620	33530
Route error	9433	8865	6667	4267
Total	49044	51740	58141	64782

Fig. 11. Routing statistics in network layer for DSR and ARM.

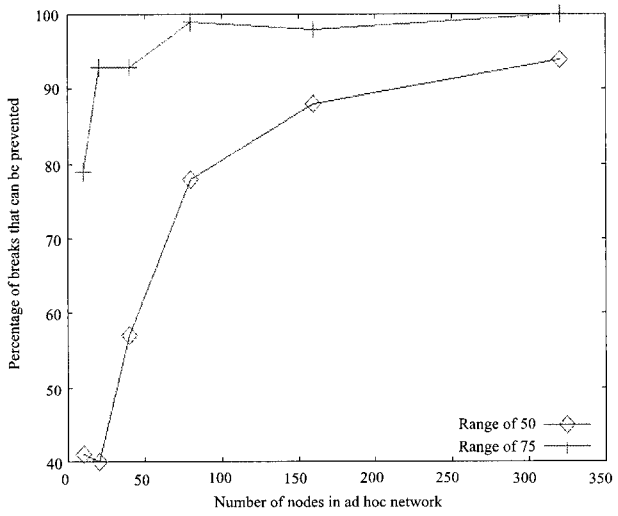


Fig. 12. The number of potential link failures that can be adverted with the expand algorithm increases as the network density increases. Results for nodes with a broadcast range of 50 and 75 are shown.

percentage of breaks that can be saved when using a broadcast radius of 50 and 75 units. Here, the expand routine can save from 30% to 100% of the breaks depending on node density.

H. Number of Nodes Involved in Flooding

An important savings from route maintenance is avoiding the cost of flooding needed to find a new route. We computed the number of nodes that would participate in a flood-based route discovery from the source in the event of a failure. We limited the range of the flood to a depth of 5, and only counted each node once (although a node may receive multiple flood messages from neighbors). As network density increases, the number of nodes participating in route-finding flood increases, as shown in Fig. 13. To summarize, as the number of nodes increases, the chance to successfully perform route maintenance increases and so does the savings.

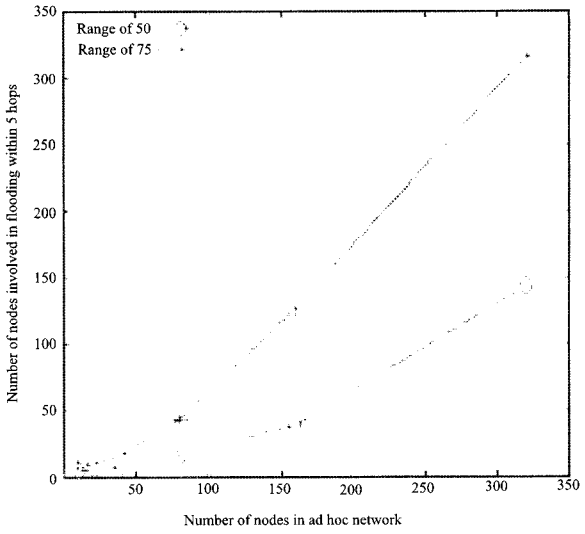


Fig. 13. Number of nodes participating in flooding with depth of five increases rapidly as the density of nodes increases. Results are shown for nodes with broadcast range of 50 and 75.

VI. CONCLUSION AND FUTURE RESEARCH

In this paper, we have proposed an efficient route maintenance protocol. By utilizing only local geographic information (now a part of some route finding algorithms), a host can anticipate its neighbor's departure and, if other hosts are available, choose a host to bridge the gap, keeping the path connected. We have presented a distributed algorithm that anticipates route failure and performs preventative route maintenance using location information to increase a route lifespan. The benefits are that this reduces the need to find new routes (which is very expensive) and prevents interruptions in service. As the density of nodes increase, the chance to successfully utilize our route maintenance approach increases, and so does the savings.

We have compared the performance of two protocols, pure dynamic source routing protocol (DSR) and DSR with ARM. The simulation results show how ARM improves the functionality of DSR by preventing the links in the route from breaking. Packets delivery ratio could be increased using ARM and achieved approximately 100% improvement. The simulations clarify also how ARM shows a noticeable improvement in dropped packets and links stability over DSR, even though there is more traffic and channel overhead in ARM. The overhead could be controlled by choosing the right value for T which depends on the node density in the network. After all, we have found that ARM is quite effective for route maintenance in the ad hoc networks because of its preventive action to save the links before we lose any packet. While the ARM-expand routine tries to add nodes in the route, the ARM-shrink routine trims any unneeded hop leading to shorter and usable path.

The proposed scheme is simple to implement and scalable since the scheme is based on distributed, local geographic information [22]. Even though ARM has been implemented on DSR in this paper, it could be added to any routing algorithm that operates based on location information. In this paper, we assumed all paths are bi-directional (since all links are assumed bi-directional). In the real world, transmission range of sending

and receiving nodes can be different. Therefore, the research based on unidirectional links would be interesting.

REFERENCES

[1] M. Ilyas, Ed., *Handbook of Ad Hoc Wireless Networks*, CRC Press LLC, 2003.

[2] C. K Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, Prentice Hall, 2001.

[3] C. Perkins and P. Bhagwat, "Highly-dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. SIGCOM'94*, 1994, pp. 234–244.

[4] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, Kluwer Academic Publishers, 1996, pp. 153–181.

[5] V. Park and M. Corson, "Highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM'97*, Apr. 1997, pp. 1405–1413.

[6] C. Perkins, E. Royer, and S. Das, "Ad hoc on demand distance vector (AODV) routing," *Internet Draft*, IETF, Oct. 1999.

[7] Y. Ko and N. Vaidya, "Location-aided routing in mobile ad hoc networks," in *Proc. ACM/IEEE MobiCom'98*, 1998, pp. 66–75.

[8] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM/IEEE MobiCom 2000*, 2000, pp. 243–254.

[9] S. Basagni, I. Chlamtac, and V. Syrotiuk, "A distance effect algorithm for mobility (DREAM)," in *Proc. ACM/IEEE MobiCom'98*, 1998, pp. 76–84.

[10] Network simulator official site for package distribution, available at http://www.isi.edu/nsnam/ns.

[11] J. Schiller, *Mobile Communications*, Addison-Wesley, 2000.

[12] W. Stallings, *Wireless Communications and Networks*, Prentice Hall, 1st ed.

[13] J. Malik, "Network simulator ns trace files analyzer," available at http://brylant.ists.pwr.wroc.pl/ jmalek/tracegraph.php.

[14] G. Lim, K Shin, S. Lee, H. Yoon, and J. S. Ma, "Link stability and route lifetime in ad hoc wireless networks," in *Proc. ICPPW 2002*, 2002, pp. 116–123.

[15] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *Int. J. Network*, vol. 30, pp. 3–30, 2001.

[16] L. Qin and T. Kunz, "Pro-active route maintenance in DSR," *ACM Mobile Computing, Commun. Rev.*, vol. 6, no. 3, pp. 79–89, July 2002.

[17] G. Aggelou and R. Tafazolli, "A Simulation analysis on reactive route repair techniques for QoS sensitive applications in mobile ad hoc networks," in *Proc. MobiHoc 2000*, Boston, USA, Aug. 2000.

[18] S. J. Lee and M. Gerla, "AODV-BR: Backup routing in ad hoc networks," in *Proc. IEEE WCNC 2000*, Sept. 2000, pp. 1311–1316.

[19] J. Li and P. Mohapatra, "LAKER: Location-aided knowledge extraction routing for mobile ad hoc networks," in *Proc. IEEE WCNC 2003*, Mar. 2003, pp. 1180–1184.

[20] I. Stojmenovic, M. Russell, and B. Vukojevic, "Depth first search and location based localized routing and QoS routing in wireless networks," *Computers and Informatics*, vol. 21, no. 2, pp. 149–165, 2002.

[21] I. Stojmenovic, "Location updates for efficient routing in ad hoc wireless networks," in *Handbook of Wireless Networks and Mobile Computing*, Wiley, 2002, pp. 451–471.

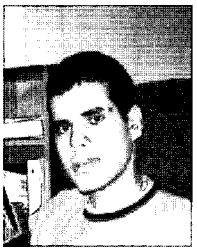
[22] I. Stojmenovic, "Position-based routing in ad hoc networks," *IEEE Commun. Mag.*, pp. 2–8, July 2002.

[23] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *ACM Wireless Networks*, vol. 6, no. 6, pp. 609–616, Nov. 2001.

[24] I. Stojmenovic, A. P. Ruhil, and D. K. Lobiyal, "Veronoi diagram and convex hull based geocasting and routing in wireless networks," in *Proc. IEEE Symp. Computers and Commun.*, Kemer-Antalya, Turkey, July 2003, pp. 51–56.

Seungjin Park received the B.E. degree in civil engineering from Hanyang University, Seoul, Korea, in 1981, the M.E. degree in civil engineering from Oklahoma State University, Stillwater, Oklahoma, in 1983, the M.S. degree in computer science from the University of Texas, Arlington, in 1988, and the Ph.D. degree in computer science from Oregon State University, Corvallis, Oregon, in 1993. From 1994 to 1999, Dr. Park was an Assistant Professor in the Department of Computer Science at Kyonggi Univeristy, Suwon, Korea. Since 1999, he is an Assistant Professor in the Department of Computer Science at Michigan Tech. University. His research interests are in the areas of mobile ad hoc networks and sensor networks.

Seong-Moo Yoo received the B.S. degree in economics from Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degree in computer science from the University of Texas at Arlington in 1989 and 1995, respectively. Since September 2001, he is an associate professor in Electrical and Computer Engineering Department of the University of Alabama in Huntsville, Huntsville, Alabama, USA. From September 1996 to August 2001, he was an assistant professor in Computer Science Department of Columbus State University in Columbus, Georgia, USA. Dr. Yoo is the conference chair of ACM Southeast Conference 2004, April, 2004, Huntsville, Alabama, USA. He was the co-program chair of ISCA 16-th International Conference on Parallel and Distributed Computing Systems (PDCS-2003), August 2003, Reno, Nevada, USA. Dr. Yoo's research interests include wireless networks, parallel computer architecture, and computer network security. Dr. Yoo is a senior member of IEEE and a member of ACM.



Mohammad Al-Shurman received the B.Sc. degree in Electrical Engineering majoring in Computer Engineering from Jordan University of Science and Technology (J.U.S.T.) in June 2000. He received his M.Sc. degree in Computer Engineering from University of Alabama in Huntsville (UAH) in August 2003. He is a Ph.D. candidate in Electrical and Computer Engineering Department, UAH. His major interests are mobile ad hoc networks, mobile routing protocols, key management, and network security.

Brian VanVoorst currently works as a principal research scientist at Honeywell Labs in Minneapolis, Minnesota. He has published in the areas of MANETs, interprocessor communication in parallel systems, distributed instrumentation systems, benchmarking, and real-time and fault-tolerant computing. Other areas of interest include sensor networks, active tag RFID systems and highly available and dependable wireless datalinks. Before coming to Honeywell, VanVoorst worked in the parallel tools group in the NAS division at NASA Ames. He received his master's degree in computer science from Michigan Technological University in 1993 and in 2003 was awarded that school's Outstanding Young Alumni Award.

Chang-Hyun Jo received his Ph.D. degree in computer science from the Oklahoma State University, in 1991. Dr. Jo was an associate professor at Kyonggi University, Korea, from 1991 to 1998. He was with University of North Dakota as an associate professor from 1998 to 2002. In 2002, he joined the Department of Computer Science at California State University Fullerton, where he is currently an associate professor. His research interests include mobile agent computing, programming languages, software engineering, ubiquitous computing, streaming technologies, networks, and security.