A Distance Routing Effect Algorithm for Mobility (DREAM)*

Stefano Basagni Imrich Chlamtac Violet R. Syrotiuk
Barry A. Woodward
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
E-mail: {basagni,chlamtac,syrotiuk,woodward}@utdallas.edu

Abstract

In this paper we introduce a new routing protocol for ad hoc networks built around two novel observations. One, called the *distance effect*, uses the fact that the greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Accordingly, the *location information* in routing tables can be updated as a function of the distance separating nodes without compromising the routing accuracy. The second idea is that of triggering the sending of location updates by the moving nodes autonomously, based only on a node's *mobility rate*. Intuitively, it is clear that in a directional routing algorithm, routing information about the slower moving nodes needs to be updated less frequently than that about highly mobile nodes. In this way each node can optimize the frequency at which it sends updates to the networks and correspondingly reduce the bandwidth and energy used, leading to a fully distributed and self-optimizing system. Based on these routing tables, the proposed directional algorithm sends messages in the "recorded direction" of the destination node, guaranteeing delivery by following the direction with a given probability. We show by detailed simulation that our protocol always delivers more than 80% of the data messages by following the direction computed, without using any recovery procedure. In addition, it minimizes the overhead used for maintaining routes using the two new principles of update message frequency and distance. Lastly, the algorithm is fully distributed, provides loop-free paths, and is robust, since it supplies multiple routes.

* This work was supported in part by the Army Research Office under contract No. DAAG55-97-1-0312.

1 Introduction

From a routing perspective, an ad hoc network is a packet radio network in which the mobile nodes perform the routing functions. Generally, routing is multi-hop since nodes may not be within the wireless transmission range of one another and thus depend on each other to forward packets to a given destination. Since the topology of an ad hoc network changes frequently, a routing protocol should be a distributed algorithm that computes multiple, cycle free routes while keeping the communication overhead to a minimum (see, e.g., [4]).

One way to classify routing protocols in ad-hoc networks is by when routes are determined. A *proactive* protocol maintains routes on a continuous basis. Thus when a sender needs to send a message, the route to the intended destination—generally, the next hop to it—is already known and can be used immediately. On the contrary, in a *reactive* approach, the sender determines a route at the time it needs to send a message, i.e., a *route discovery* phase precedes the transmission of a message.

Most of the proactive routing protocols are based on shortest path algorithms adapted to the mobile environment. This includes, e.g., the protocols presented in [2, 12], and, more recently, the Wireless Routing Protocol (WRP) introduced in [9]. In these protocols, routing tables are exchanged among neighboring nodes each time a change occurs in the topology of the network. This implies update overhead with each exchange (routes have to be recomputed according to the new information) and, since these tables are possibly large, a large part of the capacity of the network and the energy of the node is spent in their transmission. As a result, when the mobility rate of nodes is high, proactive protocols are infeasible since they cannot keep up with the changes in the topology.

An attempt to overcome the limitations of proactive protocols is instead to look for a route in an "on-demand" fashion, namely, only when it is needed to deliver a message. This is the basic idea of reactive protocols, such as Johnson and Maltz's Dynamic Source Routing (DSR) protocol [6], Park and Corson's Temporally Ordered Routing Algorithm (TORA) [10], and Perkins' Ad Hoc On-Demand Distance Vector (AODV) routing protocol [11]. In reactive protocols a control message is sent to discover (possibly more than) a route to a given destination. This kind of control message is generally shorter than the control messages used in proactive protocols, leaving more bandwidth available for the transmission of data messages. However, since a route has to be entirely discovered prior to the actual transmission of the message, a sender may experience a long delay in waiting for the route to be computed. Furthermore, there is no guarantee that the route obtained is usable, since in the meanwhile some of the nodes in the route may have moved out of transmission range. Again, the problem becomes more pronounced when the mobility rate of nodes is high, since the route discovery mechanism is not able to adapt to the variations of the speed of the nodes. Even route caching, or similar techniques, used to reduce the delay are ineffective when the mobility rate is high.

A protocol that combines both a proactive and a reactive approach has been introduced in [3]. Here, the route discovery phase is divided into an *intra zone* discovery, which involves all the nodes whose distance (i.e., number of hops) from the sender is $\leq k$ (its zone) in a proactive way, and an *inter zone* discovery, which operates between zones using a reactive approach. The appropriate choice of the zone radius k depends on the mobility rate of the nodes and on the message arrival rate (specifically, the frequency of route requests). However, this choice is static, and therefore there is no possibility to adapt to changing network conditions. Moreover, the inter zone route discovery messages may loop back into zones already queried, and this must be prevented, otherwise more overhead than flooding based approaches is incurred [5].

Whether proactive or reactive, existing routing protocols for ad hoc networks store route information similar to routing protocols for static networks—essentially, the route is stored as a sequence of nodes. In a proactive protocol the sequence is not explicit; it corresponds to a next hop table lookup at each node along the route. In a reactive protocol the result of a route discovery control message is the route given as an explicit sequence of nodes to follow. However, in the randomly changing topology of ad hoc networks, storing a route as a sequence of nodes is inadequate for reaching the destination, because the movement of any node in the sequence renders the path invalid. Thus, a new definition of routing table entry is needed.

In this paper we present a routing protocol based on a new definition of routing information. In our approach, the routing table stored at each node contains *location information* for any other node in the network (e.g., geographic coordinates that can be obtained by the use of GPS [7]). Our protocol can be considered proactive, since we define a new mechanism for the dissemination and updating of location information. When node A wants to send a message m to node B, it uses the location information for B to obtain B's *direction*, and then transmits m to all its one hop neighbors in the direction of B. Each neighbor repeats the same procedure, until B, if possible, is eventually reached. Thus, a route is sought in an on-demand fashion, like in reactive approaches.

The probability of finding B in the computed direction, relies on how the location information is disseminated through the network. In our model, each node transmits control messages bearing its current location to all the other nodes. The frequency with which these control messages are transmitted is determined by:

- considering what we call the *distance effect*: The greater the distance separating two nodes, the slower they appear to be moving with respect to each other. Thus, nodes that are far apart, need to update each others locations less frequently than nodes closer together. This is realized by associating with each control message an "age" which corresponds to how far from the sender that message travels;

- the *mobility rate*: The faster a node moves, the more often it must communicate its location. This allows each node to self optimize its dissemination frequency, thus transmitting location information only when needed and without sacrificing the route accuracy.

Since distance and mobility play a central role in our protocol, we name it the *Distance Routing Effect Algorithm for Mobility* (DREAM) protocol for ad hoc networks. Due to the new definition of routing information, we do not have to exchange large amounts of control information as in existing proactive protocols. At the same time, since no route discovery is needed, we do not suffer the associated delay typical of reactive solutions. Furthermore, DREAM achieves the following desirable properties:

- it is *bandwidth and energy efficient*: Each control message carries only the coordinates and the identifier of a node, thus being small compared to the control messages used by proactive protocols (that have to carry routing tables) and to those used by reactive protocols (that have to carry an entire route). Most importantly:

    a. The rate of control message generation is determined and optimized according to the mobility rate of each node individually.

    b. Due to the "distance effect" the number of hops (radius from the moving node) it will be allowed to travel in the network before being discarded will only depend on the relative (geographic) distance between the moving node and the location tables being updated.

  In this way the number of copies as well as the number of hops control messages will travel are both optimized (minimized) without sacrificing quality. This means that with respect to existing protocols, in DREAM more bandwidth and energy (required for transmission in each mobile node) can be used for the transmission of data messages;

- it is inherently *loop-free*, since each data message propagates away from its source in a specific direction;

- it is *robust*, meaning that the data message can reach its intended destination by following possibly independent routes;

- it is *adaptive to mobility*, since the frequency with which the location information is disseminated depends on the mobility rate.

In the next sections, we describe the mechanism of dissemination of location information, a general model to probabilistically guarantee how to find a node in a given direction, and the DREAM protocol in greater detail. The paper concludes with simulation results that show the effectiveness of our method, namely, that the protocol always delivers more than 80% of the data messages by following the direction computed,[1] and that when compared to a reactive protocol, the average end-to-end delay decreases considerably.

[1] This percentage refers to the messages delivered *directly*, i.e., to those messages that are delivered by sending them in the direction of the recipient node. If a message cannot be delivered because its intended destination cannot be found in the expected direction, our protocol provides a recovery routine that guarantees that eventually that message will be delivered.

2 Dissemination of Location Information

Consider an ad hoc network with n nodes. We assume the existence of a mechanism that allows each node to be aware of its own location (given as coordinates) with respect to a predefined positioning system (see, e.g., Global Positioning System as in [7]). These coordinates are exchanged between nodes so that each node constantly obtains location information about the other nodes in the network for routing purposes. Specifically, a *Location Table* (*LT*) is maintained at each node A that records, for each node B, its location, from which its *direction* B_ϑ and *distance* B_r can be computed. By direction we mean that B_ϑ is the angle of the polar coordinates of B on a system centered on the current position of A and by distance, we mean that B_r is the geographical distance separating A and B. The entry related to a node B, $LT(B)$, also contains $LT_\tau(B)$, the time at which the location information for B was last updated.

Since our routing protocol is based on the location table maintained at each node, care is required in order to reduce the expense of disseminating location information through the network. This is accomplished through the following simple observation: the farther two nodes are separated the less often their location table entries need updating. Intuitively, when two nodes are moving the same speed, a closer node appears to be changing more rapidly than one that is far away. We refer to this observation as the *distance effect*.

Each node, periodically broadcasts a *control packet* containing its own coordinates with respect to the specific positioning system considered. To realize the distance effect, we assign each control packet a *life time* that is based on the geographical distance the packet has traveled from its sender. A majority of the packets, will have a "short" life time: these *short lived* packets are sent at high frequency, and "die" after they have traveled through the network a short distance from their sender. Other *long lived* packets, sent less frequently, travel farther through the network, reaching the most distant nodes.[2]

When a control packet is received by a node A, the node determines how far the packet has traveled by calculating the distance d between itself and the sender of the packet. If d is greater than the life time associated with the packet, then the packet is no longer forwarded.

[2] For simplicity, we consider only two maximum "ages" for the control packets.

The frequency with which a given node broadcasts control packets is a function of the node's mobility: the more mobile the node, the more often it must disseminate its location information. The fact that most of the packets will be short lived clearly realizes the idea that the nodes closest to A are those most in need of A's location, while nodes farther away need A's location updated less often.

As a result, the further away a destination and the slower the rate of movement of the updating node, the less often a copy of the control packet will be sent. We can therefore minimize the total number of control packets in the network, while maintaining the same probability of error per route. The dissemination method described reflects the distance effect and thus maintains the same probability of routing accuracy while distributing control packets proportionately to distance and rate of movement.

Overall, this dissemination method will therefore have the following properties:

- When no movement occurs no bandwidth is wasted on control packets since control packets are initiated by moving nodes only.
- The update frequency can be optimally gauged since the decision of the update frequency lies with the moving node itself.
- The total number of control packets (and consequently related transmission energy) can be minimized since the aging of control packets captures the relative distance between the moving node and the location table updating node.

3 A Model for DREAM

The process of dissemination of location information as described in the previous section, allows us to define a model from which we can derive a probabilistic guarantee of finding a node in a given direction. When a node S needs to send a message m to a recipient node R, it refers to its LT in order to retrieve location information about R. Based on this information, S selects from among its neighbors those nodes that are in the direction of R and forwards m to them. Each of these nodes, in turn, do the same, forwarding the message to those nodes in the direction of R until R, if possible, is reached. It is thus crucial to select the neighbors of a given node in a certain direction range in such a way that it is guaranteed that R can be found with a given probability $p, 0 < p \leq 1$, following routes in that direction.

Recall, that S knows the geographical distance R_r and the angle R_ϑ of node R (indicated by r and ϑ in Figure 1, respectively), easily calculated from the location information stored in its LT table at time $LT_T(R) = t_0$. Figure 1 shows the positions of S and R at time t_0.

At some later time $t_1, t_1 > t_0$, node S wants to send a message m with node R as the recipient. S must choose among all its one hop neighbors those nodes A whose direction A_ϑ lies within the range $[\vartheta - \alpha, \vartheta + \alpha]$. The angle α must be chosen in such a way that the probability of finding R in the sector S is at least p, for a given p. The sector S is a wedge centered about the line segment connecting S and R, defined by $[\vartheta - \alpha, \vartheta + \alpha]$.

It may be seen from Figure 1 that, in the time interval from t_0 to t_1, node R, whose speed is v, cannot be anywhere outside the circle C centered on its original position with radius $x = (t_1 - t_0)v$. Here we consider that R can move in any direction β, uniformly chosen between 0 and 2π at speed v. Therefore, we want to find a minimum value for α such that the maximum distance x that R can travel in the time $t_1 - t_0$ at velocity v is within the sector S.

Clearly, α depends only on v (R's speed). If either the actual or the maximum speed of R is known to S, the value for α that guarantees that R is in the direction $[\vartheta - \alpha, \vartheta + \alpha]$ is immediately given by

$$\alpha = \arcsin \frac{v(t_1 - t_0)}{r}.$$

If the distance x that R travels is greater than the distance r separating S and R, then R can be in any direction. In this case, we must set $\alpha = \pi$.

If v is not known, and only its probability density function $f(v)$ is available, we can find a specific $\overline{\alpha}$ so that the probability of finding R in the direction $[\vartheta - \overline{\alpha}, \vartheta + \overline{\alpha}]$ is $\geq p$, for a given p, $0 < p \leq 1$. More formally, we want to determine $\overline{\alpha}$ such that

$$P(x \leq (t_1 - t_0)v) \geq p.$$

In this case, since

$$\frac{x}{\sin \alpha} = \frac{r}{\sin(\beta - \alpha)}$$

and, since when

$$\beta - \alpha = \frac{\pi}{2} \text{ it is } x = r \sin \alpha,$$

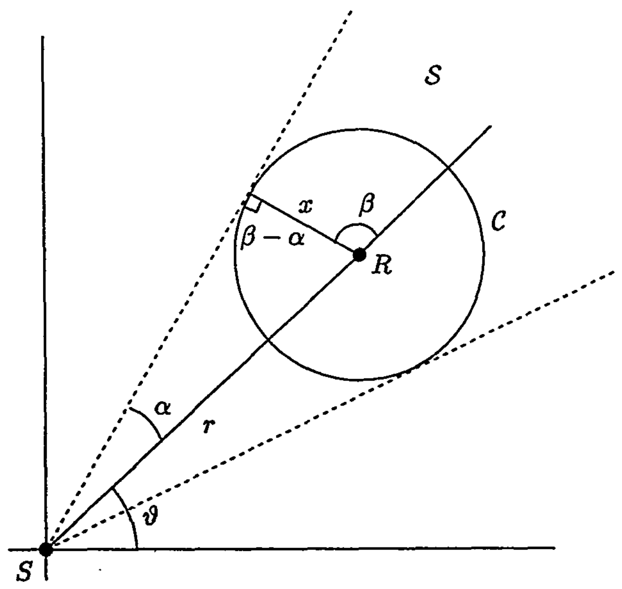

Figure 1: x is the maximum distance that the node R can travel in $t_1 - t_0$.

we want to find $\overline{\alpha}$ so that:

$$\begin{aligned} P(x \le (t_1 - t_0)v) &= P(r \sin\overline{\alpha} \le (t_1 - t_0)v) \\ &= P\left(v \ge \frac{r \sin\overline{\alpha}}{t_1 - t_0}\right) \\ &= \int_{\frac{r \sin\overline{\alpha}}{t_1 - t_0}}^{\infty} f(v)dv \end{aligned}$$

is $\ge p$.

Once the integral is computed, the needed value for α is easily obtained.

Finally, the same reasoning applies when, *given* α, we want to determine when the location information of a node R which is R_r far from S has to be updated in order to have a probabilistic guarantee to find R in the direction defined by α (i.e., we look for a value of t_1). That is, we can determine the frequency at which to disseminate location information.

4 Distance Routing Effect Algorithm for Mobility (DREAM)

In this section we describe our Distance Routing Effect Algorithm for Mobility (DREAM) based on the use of location information as informally described in the previous section. The following is a high-level algorithmic description of the procedures *Send*(m) and *Receive*(m) of DREAM executed at each node S to send or receive a message m, respectively. We use the following notation:

- m.sender, m.recipient, m.type and m.id: are the fields of the message m that contain the sender and the recipient node IDs, the type of message m (*data* or *ack*) and m's unique message identifier (at the sender). A message of type *ack* has no additional fields;
- *Time()*: returns the current value of the clock at the node executing the procedure;
- $\overline{\tau}$: is a temporal threshold value;
- *Recovery*(m): is a recovery procedure;
- *Find_Neighbors*($LT(R)$): is a function that returns a list of the one hop neighbors of S whose direction is in the range $[\vartheta-\alpha, \vartheta+\alpha]$ that depends on $LT(R)$. If no neighbor is found in the given range the value returned is **nil**. Here we assume that the maximum velocity of each node is known to all nodes;
- *Transmit*(m, list): is the procedure by which m is sent to all the neighbors specified in *list*;
- **set/clear** *Time-out*(i): sets/clears a timer associated with a message m with m.id $= i$;
- my_ID: is the identifier of the node executing the procedure.

When a node S wants to send a message m to a node R, it calls the procedure *Send*(m) below. It is assumed that the fields of m are set on entry to the procedure.

```
procedure Send(m)
begin
    R := m.recipient;
    if LT(R) = nil or Time() − LT_τ(R) > τ̄
        then Recovery(m)
        else begin
                Neighbors := Find_Neighbors(LT(R));
                if Neighbors = nil
                    then Recovery(m)
                    else begin
                            if m.type = data
                              then set Time-out(m.id);
                            Transmit(m, Neighbors)
                          end
             end
end;
```

The procedure *Send*(m) executed at node S starts by looking in S's location table to find the current expected direction of R. If no location information is available for R (indicated by $LT(R) =$ **nil**) or that information cannot be considered valid (based on $LT_\tau(R)$), then a recovery procedure must be executed in order to reach R. Here, the validity of the location information of node R is based on the time that has passed since $LT(R)$ was last updated: if this time ($Time() - LT_\tau(R)$) is greater than a certain threshold $\overline{\tau}$ then R's location information is considered obsolete. The choice of the value $\overline{\tau}$ is a crucial one: in case of "slow" moving networks it may be a constant; otherwise, it may be a function of the geographical distance of R, $LT_\tau(R)$ (and, in this case, it is stored in another field of the location table).

When the direction of R is valid, S sets a timer related to the message m and then sends m to all the one hop neighbors returned by the function *Find_Neighbors*. Notice how, given the nature of the wireless channel, the procedure *Transmit* is realized as a single transmission of m to multiple recipients. These are the neighbors of S that are within a certain direction range, as defined in the previous section. Thus, the function *Find_Neighbors*($LT(R)$) is assumed to implement the method for choosing nodes in a given direction range such that the probability of finding R in a given direction is greater than or equal to a given p, $0 < p \leq 1$. The desired probability p can be either a constant local to the function *Find_Neighbors* or it may be passed to the function as a parameter. If *Find_Neighbors* returns **nil** (i.e., if no one hop neighbor exists within the direction range specified) a recovery procedure is again in order.

The reception of a message m triggers the execution of the procedure *Receive*(m).

```
procedure Receive(m)
begin
    if my_ID = m.recipient
      then if m.type = ack
                then clear Time-out(m.id)
                else begin
                        reply.type := ack;
                        reply.id := m.id;
                        reply.sender := my_ID;
                        reply.recipient := m.sender;
                        Send(reply)
                      end
      else begin
            R := m.recipient;
            if LT(R) ≠ nil and Time() − LT_τ(R) ≤ τ̄
             then begin
                    Neighbors := Find_Neighbors(LT(R));
                    if Neighbors ≠ nil
                        then Transmit(m, Neighbors)
                  end
           end
end;
```

Upon receiving a message m, a node A first checks to see if it is the recipient of the message. If it is the intended recipient, it then looks at the message type. If the message is an acknowledgement for a data message previously sent, then the corresponding timer is cleared (and thus if it has not expired, it will no longer be considered). Otherwise, if A has just received a data message, it sends an acknowledgement to the originator of m.

If A is not the recipient of m, then it simply forwards m to all the nodes (if any) that, according to its LT, are "in the direction" of R. Notice that if, for any reason, the location information is not up to date or if no neighbor exists in the required direction, then no message is sent. This allows the timer corresponding to the message to expire which would trigger the recovery procedure.

Some comments are in order:

- If A receives more than one copy of the same message m, then, for each copy, it sends an acknowledgement: this increases the possibility that the sender receives an acknowledgement for m which in turn, increases the *robustness* of the protocol;
- The reception of an acknowledgement related to a data message for which an acknowledgement has already been received, or for which the time-out triggered the recovery procedure, has no effect;

- The time-out mechanism and the use of acknowledgements are important: the sender has to know if m has reached the recipient. Indeed, in our approach, it is possible that there is one (or even more than one) route to the recipient but its location information does not allow us to reach it. Thus, an explicit recovery procedure for this case has to be provided.

Recovery

If a sender S has no location information either available or up to date for a specific recipient, and also whenever a timer expires, an alternative method to deliver a message must be used. These situations are handled by our *Recovery*(m) procedure. Its actual implementation may vary, depending on the characteristics of the network. For instance, the message m could be partially flooded, or flooding can be used to determine a route (if any) to the recipient.

5 Simulations Results

We have simulated our DREAM protocol using MAISIE, a discrete-event simulator developed at UCLA [1], by placing $n = 30$ nodes randomly on a grid of size 100×100. For each node A, the speed is given in grid units per 100 ticks of the simulation clock (here we assume that each node has the same speed and we indicate it by V), and the transmission range tx_A is given in grid units. Two nodes A and B in the network are neighbors if the Euclidean distance d between their coordinates in the grid is less than the minimum between their transmission radii (i.e., $d(A, B) < \min\{tx_A, tx_B\}$). At every tick of the simulation clock, each node determines its direction randomly, by choosing it uniformly between 0 and 2π. Each node will then move in that direction according to its current speed. When a node reaches the grid boundary, it bounces back with an angle determined by the incoming direction. The final position of the nodes is a function of its initial position and of its current speed.

We consider three kind of messages: *control messages*, *data messages*, and *acknowledgments* (ack). Control messages carry only the location information, namely, the coordinates of the node that transmits them (here they will be the coordinates on the grid) and its identifier. Hence, they have fixed length, and their time of transmission is very short. The transmission time of an ack is short as well, since it carries no data. Data messages are considered two orders of magnitude larger than control messages (with respect to their number of bits), and therefore their transmission is accordingly longer. The rate of transmission of each node is considered uniform all over the network.

The arrival rate, namely, the frequency with which each node generates and transmits a data message, has been computed as follows. Each τ_{dm} ticks of the simulation clock each node picks a number p randomly and uniformly, $0 < p < 1$. If $p < \lambda$, then a data message m is generated and queued for transmission. The destination for m is chosen randomly and uniformly among all the other nodes of the network. In our simulations, each node has the same transmission range, which remains fixed at 40. This value guarantees good network *connectivity*, i.e., fewer than 10% of the data messages cannot be delivered to their final destination due to the lack of a physical connection (no route exists between the two nodes and therefore no routing protocol can successfully deliver messages in this case).

Each τ_{cm} ticks of the simulation clock every node A broadcasts a short lived control message with its current coordinates. This message is delivered to all those nodes whose Euclidean distance from A is less than κ grid units. Finally, one long lived control message is transmitted to all the nodes in the network for each ρ short lived control messages.

In all the simulation results presented in this paper we have chosen $0.05 < \lambda \leq 0.4$, $\tau_{dm} = 300$ and $\kappa = 40$. When the speed of a node A is 2, we have chosen $\tau_{cm} = 125$ and $\rho = 10$. Each time the speed of the nodes increases by 2 units, we decrease τ_{cm} by 30%.

Figure 2 shows that always more than 80% of the data messages delivered have reached their final destination without resorting to a recovery routine (here implemented by flooding). The three curves correspond to three different node speeds, namely, $V = 2, 4$ and 6.

In the following figures we compare DREAM with the reactive DSR protocol incorporating route caching as presented in [6]. Here, we have compared the two protocols with respect to the average *end-to-end delay* defined as follows: if M is the set of all the messages delivered, and for each $m \in M$, t^s_m and t^r_m correspond to the time when m is generated at the sender and queued for transmission, and received at the destination, respectively, then the average end-to-end *delay* is computed as follows:

$$\frac{1}{|M|} \sum_{m \in M} (t^r_m - t^s_m) ,$$

where $|M|$ is the total number of messages transmitted. Figures 3 and 4 show that the average end-to-end *delay* of the DSR protocol is from 25% to 250% larger than the delay obtained by DREAM protocol (the two figures

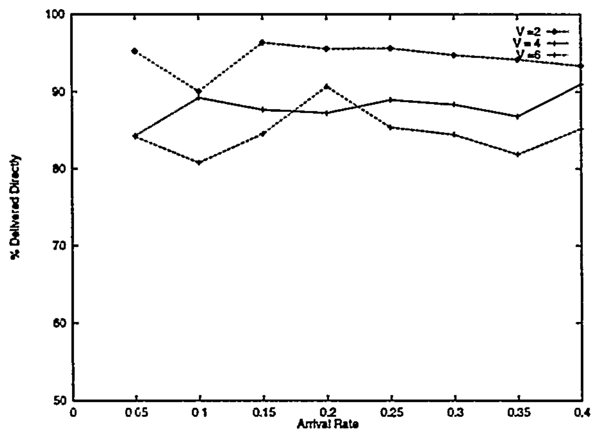

Figure 2: Percentage of messages delivered without resorting to the recovery procedure, when the nodes have three different speeds.

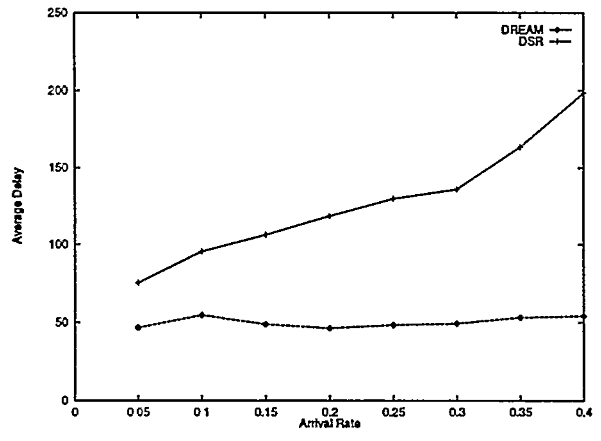

Figure 3: Average delay vs. arrival rate for DREAM and DSR when each node has speed $V = 2$.

correspond to two different node speeds, $V = 2$ and 6). Furthermore, the average delay for DREAM remains essentially constant for all arrival rates.

The confidence level of all our results is 95%, and their precision is within 5%.

6 Conclusions

In this paper we have presented a new directional routing protocol for ad hoc networks using a novel mechanism for the dissemination of location information. The proposed solution can be used to minimize the amount of bandwidth and transmission power used to maintain routing tables without penalizing the accuracy of the

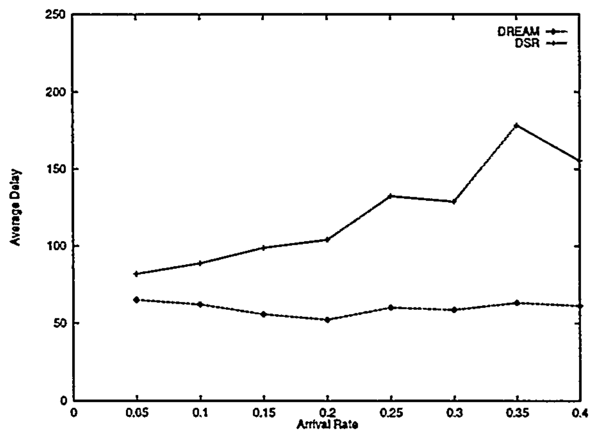

Figure 4: Average delay vs. arrival rate for DREAM and DSR when each node has speed $V = 6$

routing tables. Based on these routing (location) tables a probabilistic method for selecting the direction in which a given node may be found was proposed. Simulation results showed that with over 80% probability this method can find a route (if any exists) to a given node in the direction computed by DREAM while the average end-to-end delays with respect to the DSR reactive protocol are substantially lower. Finally, the DREAM protocol provides loop-free routes, and is robust in providing multiple routes to a given destination.

References

[1] BAGRODIA, R. L., AND LIAO, W.-T. Maisie: a language for the design of efficient discrete-event simulations. *IEEE Transactions on Software Engineering 20*, 4 (April 1994), 225–238.

[2] CHENG, C., RILEY, R., KUMAR, S. P. R., AND GARCIA-LUNA-ACEVES, J. J. A loop-free extended bellman-ford routing protocol without bouncing effect. *Computer Communication Review 19*, 4 (September 1989), 224–236.

[3] HAAS, Z. J. A new routing protocol for the reconfigurable wireless network. In *Proceedings of the 1997 IEEE 6th International Conference on Universal Personal Communications, ICUPC'97* (San Diego, CA, 12–16 October 1997), pp. 562–566.

[4] HAAS, Z. J. Panel report on ad hoc networks—Milcom'97. *Mobile Computing and Communications Review, MC^2R, a publication of the ACM SIGMOBILE 2*, 1 (January 1998), 15–18.

[5] HAAS, Z. J., AND PEARLMAN, M. R. The zone routing protocol (ZRP) for ad hoc networks. INTERNET DRAFT—Mobile Ad hoc NETworking (MANET) Working Group of the Internet Engineering Task Force (IETF), November 1997. To be considered Work in Progress. See also [8].

[6] JOHNSON, D., AND MALTZ, D. A. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, T. Imielinski and H. F. Korth, Eds. Kluwer Academic Publishers, Dordrecht, The Netherlands, February 1996, ch. 5, pp. 153–181.

[7] KAPLAN, E. D., Ed. *Understanding GPS : principles and applications.* Artech House, Boston, MA, 1996.

[8] MACKER, J. P., AND CORSON, M. S. Mobile ad hoc networking and the IETF. *Mobile Computing and Communications Review, MC^2R, a publication of the ACM SIGMOBILE 2*, 2 (April 1998), 9–12.

[9] MURTHY, S., AND GARCIA-LUNA-ACEVES, J. J. An efficient routing protocol for wireless networks. *ACM/Baltzer Journal on Mobile Networks and Applications, MANET 1*, 2 (October 1996), 183–197.

[10] PARK, V., AND CORSON, M. S. A highly adaptive distributed algorithm for mobile wireless networks. In *Proceedings of the IEEE INFOCOM'97* (Kobe, Japan, 7–11 April 1997).

[11] PERKINS, C. E. Ad hoc on-demand distance vector (AODV) routing. INTERNET DRAFT—Mobile Ad hoc NETworking (MANET) Working Group of the Internet Engineering Task Force (IETF), 20 November 1997. To be considered Work in Progress. See also [8].

[12] PERKINS, C. E., AND BHAGWAT, P. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *Computer Communication Review 24*, 4 (October 1994), 234–244.