# Gossip-Based Ad Hoc Routing

Zygmunt Haas Joseph Y. Halpern Li Li
School of Electrical and Computer Engineering/Department of Computer Science
Cornell University
haas@ece.cornell.edu {halpern,lili}@cs.cornell.edu

*Abstract*— **Many *ad hoc* routing protocols are based on (some variant of) flooding. Despite various optimizations, many routing messages are propagated unnecessarily. We propose a gossiping-based approach, where each node forwards a message with some probability, to reduce the overhead of the routing protocols. Gossiping exhibits bimodal behavior in sufficiently large networks: in some executions, the gossip dies out quickly and hardly any node gets the message; in the remaining executions, a substantial fraction of the nodes gets the message. The fraction of executions in which most nodes get the message depends on the gossiping probability and the topology of the network. In the networks we have considered, using gossiping probability between 0.6 and 0.8 suffices to ensure that almost every node gets the message in almost every execution. For large networks, this simple gossiping protocol uses up to 35% fewer messages than flooding, with improved performance. Gossiping can also be combined with various optimizations of flooding to yield further benefits. Simulations show that adding gossiping to AODV results in significant performance improvement, even in networks as small as 150 nodes. We expect that the improvement should be even more significant in larger networks.**

## I. Introduction

An *ad hoc network* is a multi-hop wireless network with no fixed infrastructure. Rooftop networks and sensor networks are two existing types of networks that might be implemented using the ad hoc networking technology. Ad hoc networks can be usefully deployed in applications such as disaster relief, tetherless classrooms, and battlefield situations.

In ad hoc networks, the power supply of individual nodes is limited, wireless bandwidth is limited, and the channel condition can vary greatly. Moreover, since nodes can be mobile, routes may constantly change. Thus, to enable efficient communication, robust routing protocols must be developed.

Many ad hoc routing protocols have been proposed. Some, such as LAR [13], GPSR [12], and DREAM [1] assume that nodes are equipped with GPS hardware and thus know their locations; others, such as DSR [11], AODV [18], ZRP [9], and TORA [17], do not make this assumption. Essentially all protocols that do not use GPS (and some that do, such as LAR and DREAM) make use of flooding, usually with some optimizations.

Despite the optimizations, in routing protocols that use flooding, many routing messages are propagated unnecessarily. In this paper, we show that *gossiping*—essentially, tossing a coin to decide whether or not to forward a message—can be used to significantly reduce the number of routing messages sent.

Our key observation is that gossiping exhibits bimodal behavior; this essentially follows from results in a branch of mathematics known as *percolation theory* [7], [15]. Let the gossip probability be $p$. Then, in sufficiently large "nice" graphs, there are fractions $\theta^S(p)$ and $\theta^R(p)$ such that the gossip quickly dies out in $1-\theta^S(p)$ of the executions and, in almost all of the fraction $\theta^S(p)$ of the executions where the gossip does not die out, a fraction $\theta^R(p)$ of the nodes get the message. Moreover, in many cases of interest, $\theta^R(p)$ is close to 1. Thus, in almost all executions of the algorithm, either hardly any nodes receive the message, or most of them do. Ideally, we could make the fraction of executions where the gossip dies out relatively low while also keeping the gossip probability low, to reduce the message overhead. The goal of this paper is to investigate the extent to which this can be done. Our results show that, by using appropriate heuristics, we can save up to 35% message overhead compared to flooding. Furthermore, adding gossiping to a protocol such as AODV not only gives improvements in the number of messages sent, but also results in improved network performance in terms of end-to-end latency and throughput. We expect that the various optimizations applied to flooding by other protocols (for example, the cluster-based scheme of [16]) can also be usefully combined with gossiping to get further performance improvements.

We are certainly not the first to use gossiping in networking applications. For example, it has been applied in networked databases to spread updates among the nodes [6] and to multicasting [2]. However, in almost all of the earlier work on gossiping, it is assumed that any node in the network can send a message to any other node, either because there is a direct link to that node or a route to that node is known. Gossiping proceeds by choosing some set of nodes at random to which to gossip. We do not have the luxury of being able to make such an assumption in the context of ad hoc networks. Our problem is to find routes to different nodes.

In an ad hoc network, if a message is transmitted by a node, due to the nature of radio communications, the message is usually received by all the nodes one hop away from the sender. Because of the fact that wireless resources are expensive, it makes sense to take advantage of this physical-layer broadcasting feature of the radio transmission. In our gossiping protocol, we control the probability with which this physical-layer broadcast is sent.

There has been some recent work on applying gossiping in ad hoc networks, but the focus and thus the techniques used have been very different from our work. Vahdat and Becker [22] apply gossiping to ad hoc unicast routing. However, their usage of gossiping is very different from ours. In their work, they try to ensure that messages are eventually delivered even if there is no connected path between the source and the destination at any given point in time. As long as there exists a path using communication links at some point in time, messages can be delivered through a random pair-wise exchanges among mobile hosts. Their techniques are not intended for (and would not perform well in) our setting, where we are trying to find routes that we assume exist (because network partition is a rare event). Chandra et al. [4] use a gossiping mechanism to improve multicast reliability in ad hoc networks; they do not use gossiping to reduce the number of messages sent. Indeed, they start with an arbitrary, possibly unreliable, multicast protocol to multicast a message. They then use gossiping (under the assumption that routes are known) to randomly exchange messages between nodes in order to recover lost messages. Heinzelman et al. [10] have applied gossiping in data dissemination in wireless sensor networks, using techniques similar in spirit to those of [22]. Again, the setting and results are quite different from ours. Ni et al. [16] propose five different approaches to reduce broadcast redundancy. One of them (briefly mentioned in a few sentences) is gossiping. However, they do not study the properties of gossiping nor do they consider heuristics for dealing with problems introduced by gossiping in realistic ad hoc network topologies. Their experiments do show that, in a 100-node network, using gossiping can save messages.

The rest of this paper is organized as follows: Section II discusses the basic bimodal effect in more detail. Section III provides experimental evidence of the bimodal effect in networks of reasonable size, and also gives a sense of how the probability varies with the average degree of the network and initial conditions. Section IV presents a number of heuristics that should improve the performance of gossiping in networks of interests, and investigates the extent to which they do so experimentally. Section V shows that gossiping can help in practical settings, by considering the effect of adding gossiping to AODV. We show by simulation that even in networks with 150 nodes only, adding gossiping to AODV can result in significant performance improvements on all standard metrics. We expect that this improvement will be even more significant in larger networks. Section VI concludes our paper.

## II. The Bimodal Behavior of Gossiping

Since flooding is a basic element in many of the ad hoc routing protocols, as mentioned in Section I, we start by comparing gossiping to flooding.

Our basic gossiping protocol is simple. A source sends the route request with probability 1. When a node first receives a route request, with probability $p$ it broadcasts the request to its neighbors and with probability $1 - p$ it discards the request; if the node receives the same route request again, it is discarded. Thus, a node broadcasts a given route request at most once. This simple protocol is called GOSSIP1($p$).

GOSSIP1 has a slight problem with initial conditions. If the source has relatively few neighbors, there is a chance that none of them will gossip, and the gossip will die. To make sure this does not happen, we gossip with probability 1 for the first $k$ hops before continuing to gossip with probability $p$. We call this modified protocol GOSSIP1($p, k$).[1]

The performance of GOSSIP1($p, k$) clearly depends on the choice of $p$ and $k$. Clearly, GOSSIP1(1,1) is equivalent to flooding. What happens in general? That depends in part on the topology of the network (particularly its average degree), the gossip probability $p$, and the initial conditions (as determined by $k$). If we think of gossiping as spreading a disease in an epidemic, this simply says that the likelihood of an epidemic spreading depends in part on how many people each person can infect (the degree), the likelihood of the infection spreading (the gossip probability), and how many people are initially infected.

As we said in the introduction, gossiping and, in particular, the performance of GOSSIP1($p$,0) (that is, the scenario where even the source gossips with probability $p$) has been well studied in the work on percolation theory [7], [15]. Quite a few types of networks have been studied in the literature. The following theorem gives a sense of the type of results that have been proved.

*Theorem II.1:* For all $p \geq 0$, for all infinite regular graphs $G$, and for almost all (i.e., a measure 1 subset)

[1] Of course, the fact that gossiping has difficulties if a node has relatively few neighbors is true not just initially. We return to this point in the next section when we discuss optimizations.

of the infinite random graphs $G$ constructed as above, if GOSSIP1($p$,0) is used by every node to spread a message, then there is a probability $\theta_0(p) < 1$ such that the message reaches infinitely many nodes is $\theta_0(p)$. Moreover, the probability that a node receives the message and forwards it in an execution where the message reaches infinitely many nodes is also $\theta_0(p)$.

Note that the probability of a message dying out (i.e., not spreading to infinitely many nodes) is taken over the executions of the algorithm. That is, the theorem says that if we execute the algorithm repeatedly, the probability that a message does not die out in any given execution is $\theta_0(p)$. On the other hand, the second $\theta_0(p)$ talks about the probability that a node receives and forwards the message in a given execution of the algorithm. The intuition behind the equality of the above two probabilities is easy to explain. A gossip initiated by a source $n_0$ dies out if there is a set of nodes $N$ that disconnects $n_0$ from the rest of the graph; that is, every infinite path starting at $n_0$ must go through a node in $N$. Thus, $\theta_0(p)$ is the probability that there is no disconnecting set $N$ such that none of the nodes in $N$ forward the message. (Note that $N$ could consist of the singleton node $n_0$ itself.) Similarly, the probability that a random node $n$ does not receive and forward the message is precisely the probability that there is a set $N'$ such that $N'$ disconnects $n$ from $n_0$ and none of the nodes in $N'$ forwards the message. In a regular graph or a random graph of the type considered here, these probabilities are clearly the same.

It follows from these results that, in an execution where the message does not die out, the probability that a random node receives the message is $\theta_0(p)/p$, since receiving the message is independent of forwarding it. Thus, in terms of the notation used in the introduction, $\theta^S(p) = \theta_0(p)$ and $\theta^R(p) = \theta_0(p)/p$.

Let $\theta_k^S(p)$ be the probability that a message reaches infinitely many nodes if GOSSIP1$(p,k)$ is used. It is easy to see that $\theta_1^S(p) = \theta_0(p)/p$, since the probability that the message reaches infinitely many nodes using GOSSIP1$(p,1)$ is precisely the probability that a message reaches infinitely many nodes using GOSSIP1$(p,0)$ given that the source actually gossips. However, note that the probability that a node receives and forwards a message if GOSSIP1$(p,k)$ is used, given that the message does not die out, is still $\theta_0(p)$. That is, the probability that a node receives the message is independent of the choice of $k$. On the other hand, it is not hard to see that if each node learns the network topology in a zone of radius $k$ (so that it can route a message directly to any node in its zone), then the probability that a node receives and forwards a message given that the message does not die out is $\theta_k(p)$.

All these results are for infinite graphs. It is not hard to show that essentially the same results hold for finite graphs, except possibly near the boundary. In sufficiently large finite graphs, there will be two types of executions: those where hardly any node gets the message, and those where the message makes it all the way to the boundary. It follows easily from the Central Limit Theorem that, in sufficiently large graphs, in almost all executions where the gossip does not die out, a fraction $\theta_0(p)/p$ nodes will get the message. That is, we expect the bimodal behavior: either hardly any nodes get the message, or a fraction $\theta_0(p)/p$ receive the message. As we shall see, in cases of interest, $\theta_0(p)$ is quite close to $p$. Thus, in almost all executions of the algorithm in sufficiently large graphs, either hardly any nodes receive the message, or most do.

This leads to a number of obvious questions:

- How large is "sufficiently large"?
- What is the behavior of $\theta_k(p)$ for different graphs of interest?
- What can be done to improve the performance of gossiping in realistic settings?

We investigate these questions in the next two sections.

## III. GOSSIPING IN FINITE NETWORKS

We did a number of experiments to investigate the behavior of gossiping. We summarize some of the more interesting results here. We assumed an ideal MAC layer for these experiments because we wanted to decouple the effect of the MAC layer from the effect of gossiping. (When we consider more realistic scenarios in Section V, we use the IEEE 802.11 MAC layer.) We first study regular networks, since they allow us to easily analyze how GOSSIP1 behaves with respect to different parameters, such as the gossip probability, network size, and node degree, without other complicating factors. As we shall see, the behavior in regular graphs seems quite indicative of the behavior of gossiping in general. We then study random networks constructed as follows: Nodes are placed at random on a two-dimensional area; an edge is placed between any pair of nodes less than a fixed distance $d$ apart. This type of random graph seems appropriate for modeling a number of applications involving ad hoc networks. Nodes have a limited amount of power, and so can communicate only with reasonably close nodes. The random placement can be viewed as modeling features such as the random mobility of nodes and the random placement of sensors in a large region.

Our first set of experiments involves "medium-sized" networks, with 1000 nodes. We start by considering a $20 \times 50$ grid (i.e., a regular graph of degree 4). We focus on GOSSIP1$(p,4)$, since taking $k = 4$ produces a reasonable tradeoff. (We report below on the effect of varying $k$.) The results depend in part on where we place the route request source. As we would expect from the theoretical arguments, the location of the source node does not affect the fraction of nodes receiving the message. However, it does affect the number of executions in which the gossip dies out. The number of executions in which the gossip does not die out is higher for a more central node, and lower for a corner node. We report results here for the case where the route request source is at the left boundary of row 10. Our experiments show that, on average, the performance is somewhat better than the reported results. The results are illustrated in Figure 1. Notice that GOSSIP1(.72,4) on the grid ensures that almost all nodes get the message, except for a slight dropoff at distance greater than 50. This dropoff is a boundary effect; we discuss it in more detail below. Note that the graph in Figure 1(a) represents an average of 120 executions of the protocol. With gossip probability .72 for this grid size, in almost all the executions of the algorithm, almost all nodes get the message.

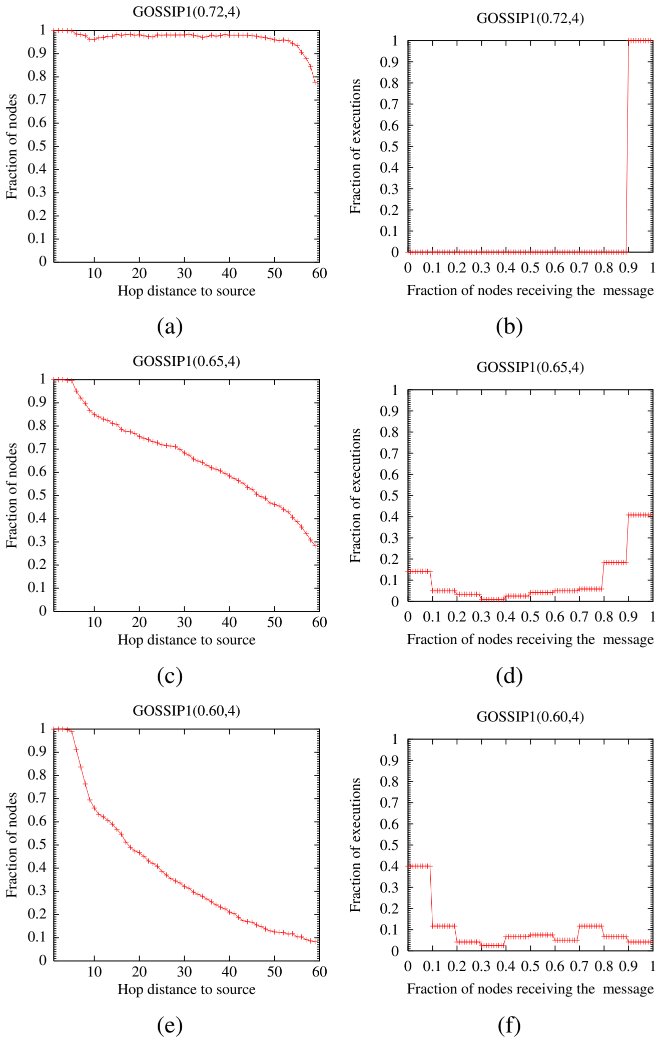



Fig. 1. The behavior of gossiping on a $20 \times 50$ grid.

The situation changes significantly if the gossip probability is even a little less than .7. For example, the average performance of GOSSIP1(.65,4) is shown in Figure 1(c). As the graph shows, at distance 40, on average 58% of the nodes got the message. However, in this case, the graph is somewhat misleading. The averaging is hiding the true behavior. As we would expect from Theorem II.1, there is bimodal behavior. This is illustrated in Figure 1(d). If we consider nodes at distance 15–45 (so as to ignore initial effects and boundary effects), in 14% of the executions, fewer than 10% of the nodes get the message; in 19% of the executions, fewer than 20% of the nodes get the message; in 59% of the executions, more than 80% of the nodes get the message; and in 41% of the executions, more than 90% of the nodes get the message.

If we lower the gossip probability further, we get the same bimodal behavior; all that changes is the fraction of executions in which all nodes and no nodes get the message. The dropoff is fairly rapid. For example, Figure 1(e) and (f) describe the situation for GOSSIP1(.60,4). By the time we get to probability .6 on the grid, in only 4% executions of the algorithm is it the case that more than 90% of the nodes get the message; in only 11% of the executions do more than 80% of the nodes get the message; and in over 50% of the executions, fewer than 20% of the nodes get the messages.

We also investigated the effect of the degree of the network on gossiping. Not surprisingly, increasing the degree makes it better and decreasing it makes it worse. In a $20 \times 50$ regular network of degree 6, it suffices to gossip with probability .65 to ensure that almost all nodes get the message in almost all executions; with gossip probability .6, we start to see some dropoff. (Again, this is average behavior; the underlying situation is bimodal.) On the other hand, for a $20 \times 50$ regular network of degree 3, we need to gossip with probability .86 to ensure that almost all nodes get the message in all executions.

While easy to study, regular graphs are not typical of the topology we expect in ad hoc networks. Random graphs are a somewhat better model. We considered two families of random graphs. In the first, we randomly placed 1000 nodes in a $7500m \times 3000m$ rectangular region, where a node can communicate with another node if it is no more than 250 meters away. This results in a network with average degree 8. The results are illustrated in Figure 2.

The results are qualitatively similar to those on the grid, as we would expect. Indeed, the bimodal effect is particularly pronounced with GOSSIP1(.65,4), as shown in Figure 2(d). If we consider nodes at distance 15–35, Figure 2(d) shows, in 20% of the executions, fewer than 10%

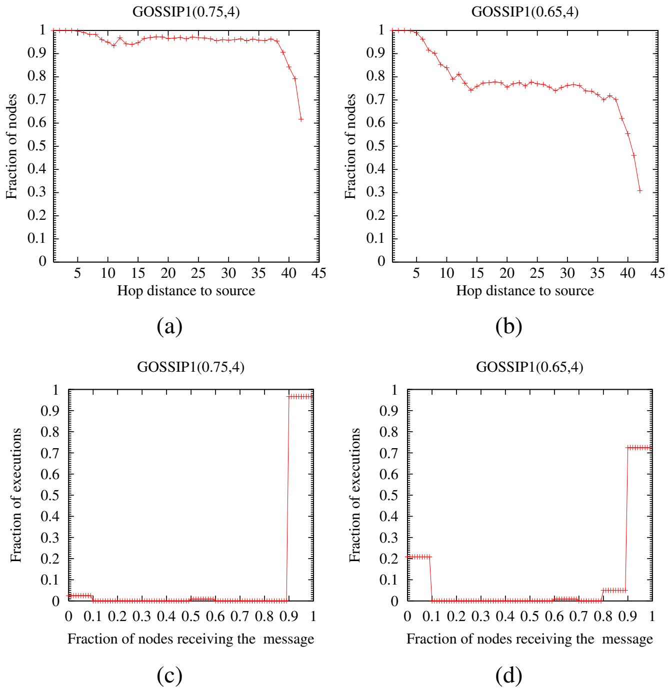



Fig. 2. Gossiping on a random network of average degree 8.

of the nodes get the message; in 70% of the executions, over 90% of the nodes get the message, and in 75% of the executions, over 80% of the nodes get the message.

To consider what happens with a higher-degree network, we also placed 1200 nodes at random in the same rectangular region; this results in a network with average degree 10. In this network, it suffices to gossip with probability .65 to ensure that almost all nodes get the message in almost all executions.

All the graphs above show a marked dropoff in probability for nodes that are close to the boundary. This is not just an effect of averaging; this dropoff occurs in almost all executions of the algorithm. The dropoff is due to two related boundary effects.

1. Distant nodes have fewer neighbors, since they are close to the boundary.
2. Nodes at distance $d$ from the source may well receive message due to "back-propagation" from nodes at distance $d' > d$ that get the message. Such back-propagation is not possible for boundary nodes.

We discuss some techniques to deal with this dropoff in Section IV-D.

We did one last set of experiments to better evaluate $\theta_k(p)$. In these experiments, we used 1,000,000 nodes on a $1000 \times 1000$ grid, and placed the source at the center of row 10. This is far enough away from the boundary to avoid significant boundary effects.[2] The results of using GOSSIP4($p,k$) for particular values of $p$ are illustrated in Figure 3. As these results show, the bimodal effect is very marked by the time we get to such a large network, and begins to closely approximate the results expected from the theorem. Figure 4 shows how $\theta_4^S(p)$ varies with $p$. As we can see, if $p$ is below .59, then the gossip dies out in almost all executions. $\theta_4^S(p)$ then increases very rapidly, going from 0 at .59 to almost 1 at .65. (The rapid increase in the case of infinite graphs follows from a deeper mathematical analysis, and has been discussed in the percolation theory literature.)

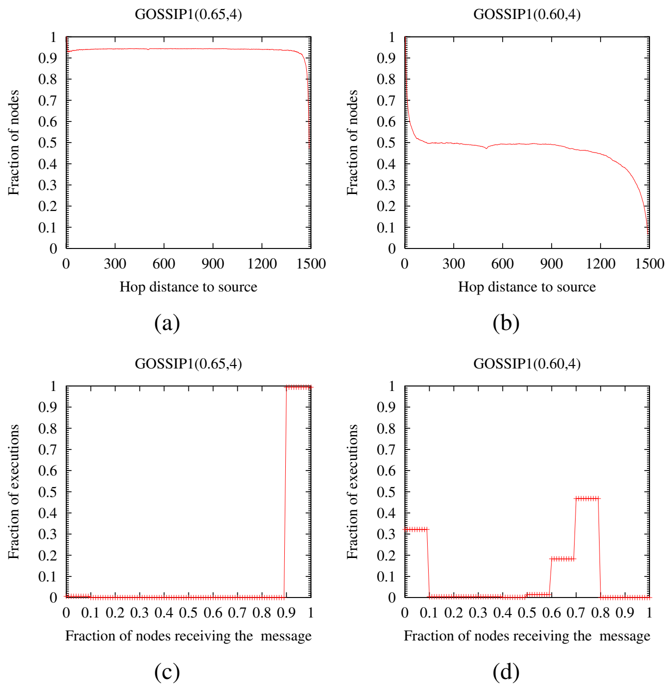



Fig. 3. The behavior of gossiping on a 1000x1000 grid.

Finally, we considered how $\theta_k^S(p)$ and $\theta_k^R(p)$ varied with $k$ for a fixed value of $p$. As theory predicts, $\theta_k^R(p)$ does not change at all with $p$. There is some effect on $\theta_k^S(p)$. Of course, since $\theta_1^S(p) = \theta_0^S(p)/p$, there is a significant jump as $k$ goes from 0 to 1. As $k$ increases beyond 1, there is an increase in $\theta_k^S(p)$, but it is not so significant. For example, $\theta_1^S(.65) = .95$, $\theta_2^S(.65) = .98$, and $\theta_5^S(.65) = 1$; similarly, $\theta_1^S(.6) = .53$, $\theta_4^S(.5) = .67$, and $\theta_{10}^S = .73$.

[2] Actually, there are boundary effects for values of $p$ very close to, but above, .59, which is the threshold below which the gossip will almost surely die. This is unavoidable. It can be shown that there are nontrivial boundary effects for values of $p$ very close to .59 no matter where we place the source.

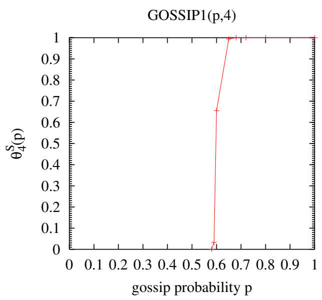



Fig. 4. $\theta_4^S(p)$ as a function of the gossip probability $p$ on a 1000x1000 grid.

## IV. Heuristics to Improve the Performance of Gossiping

The results of the previous section suggest an obvious way that gossiping can be applied in ad hoc routing. Rather than flooding, we use GOSSIP1($p$,$k$) with $p$ sufficiently high to guarantee that almost all nodes will receive the message in almost all executions. We can practically guarantee that the destination node receives the message, while saving a fraction $1 - p$ of messages. In cases of interest, where the threshold probability seems to be about .65–.75, this means we can ensure that all nodes get the message using 25–35% fewer messages than flooding.

The basic gossiping scheme can be optimized in a number of ways, using ideas that have been applied to flooding and ideas specific to gossiping. We discuss some optimizations in the remainder of this section.

### A. A two-threshold scheme

In many cases of interest, a gossip protocol is run in conjunction with other protocols. If the other protocols maintain fairly accurate information regarding a node's neighbors, we can make use of this information to improve the performance of GOSSIP1 further by a simple optimization.

In a random network, unlike the grid, a node may have very few neighbors. In this case, the probability that none of the node's neighbors will propagate the gossip is high. To prevent this, we consider a protocol with four parameters, $p_1$, $k$, $p_2$, and $n$. As in GOSSIP1, $p_1$ is the typical gossip probability and $k$ is the number of hops with which we start gossiping with probability 1. The new features are $p_2$ and $n$; the idea is that the neighbors of a node with fewer than $n$ neighbors gossip with probability $p_2 > p_1$. That is, if a node has fewer than $n$ neighbors, it instructs its immediate neighbors to broadcast with probability $p_2$ rather than $p_1$. Call this modified protocol GOSSIP2($p_1, k, p_2, n$). GOSSIP2 is not of interest in regular networks. However, in random networks which typically have some sparse regions, it can have a significant impact. For example, for the random network with average degree 8 first considered in Figure 2, GOSSIP2(0.6,4,1,6) has better performance than GOSSIP1(0.75,4), as shown in Figure 5, while using 4% less messages than GOSSIP1(0.75,4). Only when $p \geq 0.8$ does GOSSIP1($p$, 4) begin to have the same performance as GOSSIP2(0.6,4,1,6); however, GOSSIP1(0.8,4) uses 13% more messages than GOSSIP2(0.6,4,1,6).

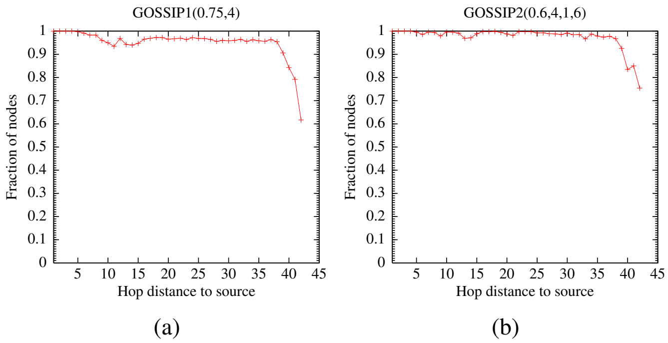



Fig. 5. Gossiping with two thresholds vs. one on a random network of average degree 8.

There may be other combinations of parameters for GOSSIP2 that give even better performance; we have not checked exhaustively. The key point is that using a higher threshold for successors of nodes with low degree seems to significantly improve performance.

### B. Preventing premature gossip death

As we have seen, the real problem with gossiping is that, if we gossip with too low a probability, the message may "die out" in a certain fraction of the executions. Measures can be taken to prevent this (for example, having successors of nodes with low degree gossip with a higher probability) but, unfortunately, there is no way for a node to know if a message is dying out. Nevertheless, a node may get some clues. One such clue is not getting too many copies of the message. Suppose that a node $x$ got the message but does not broadcast it because its coin toss landed "tails". Further suppose that $x$ has $n$ neighbors. If the message does not die out, then it would expect that all of its neighbors would get the message as well, and thus, if the gossip probability is $p$, it should get roughly $pn$ messages from its neighbors. If it gets significantly fewer than $pn$ within a reasonable time interval, then this is a clue that the message is dying out.

This suggests the following optimization of GOSSIP1 and GOSSIP2. If a node with $n$ neighbors receives a message, does not broadcast it, but then does not receive the message from $m$ neighbors within a reasonable timeout period, it broadcasts the message to all its neighbors. The obvious question here is what $m$ should be. If $m$ is chosen too large, then we may end up with too many messages. Our experiments show that we actually get the most significant performance improvement by taking $m = 1$. Let GOSSIP3($p, k, m$) be just like GOSSIP1($p, k$), except that a node broadcasts a message if it initially got it and did not broadcast it (because its coin landed tails), but it did not get the message from at least $m$ other nodes after first getting it. (The choice of timeout period can be taken quite small. We discuss this issue in details in Section V.) It may seem that such rebroadcasting can significantly effect the latency of the message. However, as the experiments discussed below show, if the parameters are chosen correctly, latency is not a problem at all.

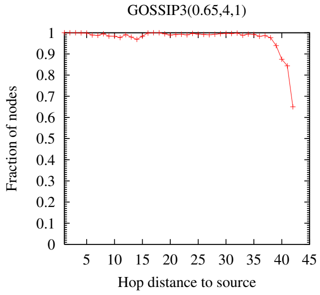



Fig. 6. GOSSIP3 on a random network of degree 8.

As Figure 6 shows, the performance of GOSSIP3(0.65,4,1) is even better than that of GOSSIP1(0.75,4). However, GOSSIP3(0.65,4,1) sends only 67% of the messages sent by flooding. By way of contrast, GOSSIP1(0.75,4) sends 75% of the messages sent by flooding. Thus, we get better performance using GOSSIP3 while sending 8% fewer messages.

To examine the effect of GOSSIP3 on latency, we recorded the number of timeout intervals a message experienced, using a variable $L$ which was augmented every time a message was forwarded after a timeout. Among all the messages sent by GOSSIP3(0.65,4,1), only 2% have $L \geq 1$. Among these messages with $L \geq 1$, 95% of them have $L \leq 2$. Thus, it seems latency is not a problem here.

### C. Retries

The bimodal distribution observed in the use of gossiping can be viewed as a significant advantage. Once a route is found, acknowledgments are propagated back to the source along the route, so the source knows the route. If a route is not found within a certain timeout period, there are two possibilities: either there is no route at all, or the protocol did not detect it. Our focus is on networks that are sufficiently well connected that there typically is a route. However, when using a gossiping protocol, there is always a possibility that a route will not be found even if it exists. Of course, there is a simple solution to this problem: simply retry the protocol. Thus, for example, the probability of finding a route within two attempts to a node at distance 25 using GOSSIP1(.65,4) in the random network with average outdegree 8 is .95: the probability of a node not receiving a message in any given execution of the protocol is .23, and executions are independent.

With retries, the bimodal message distribution works significantly to our benefit. As we observed, with GOSSIP1(.65,4), in 72% of the executions, almost all nodes get the message. If we pick a destination at random, in those executions where almost all nodes get the message, the destination is likely to get the message and a retry will not be necessary. On the other hand, in those executions where hardly any nodes got the message, a retry will probably be necessary. However, such failing gossip attempts do not involve too many transmissions, since most nodes do not get the message.

Of course, retries increase latency, even if they do not significantly increase the number of messages sent. This is especially true in large networks, where the timeout period will have to be large so as to allow the message to propagate throughout the network. However, even here, the bimodal distribution can be used to advantage to decrease the retry latency. Note that each message must keep track of the number of hops it has taken. We can modify the algorithm so as to require that any node that receives a message with, say, 15 hop counts, forwards an acknowledgment to the sender along that route with some probability. (The probability can be chosen so that the sender receives an expected number of, say, five acknowledgments if almost all nodes get the message.) Because of the bimodal distribution, if the sender does receive several acknowledgments, then it can be fairly confident that the execution is one in which almost all nodes are getting the message. On the other hand, if it does not receive several acknowledgments, it is likely that the execution is one in which hardly any nodes get the message, and it should resend. This shows that we can bound the latency of retry, independent of the network size.

### D. Zones

One of the best-known optimizations to flooding is the *zone routing protocol* (ZRP) [9]. In ZRP, each node $u$ maintains a so-called *zone*, which consists of all the nodes that are at most $\rho$ hops away from $u$, for some appropriately chosen *zone radius* $\rho$. A node that is exactly $\rho$ hops away from $u$ is called a *peripheral node* of $u$.

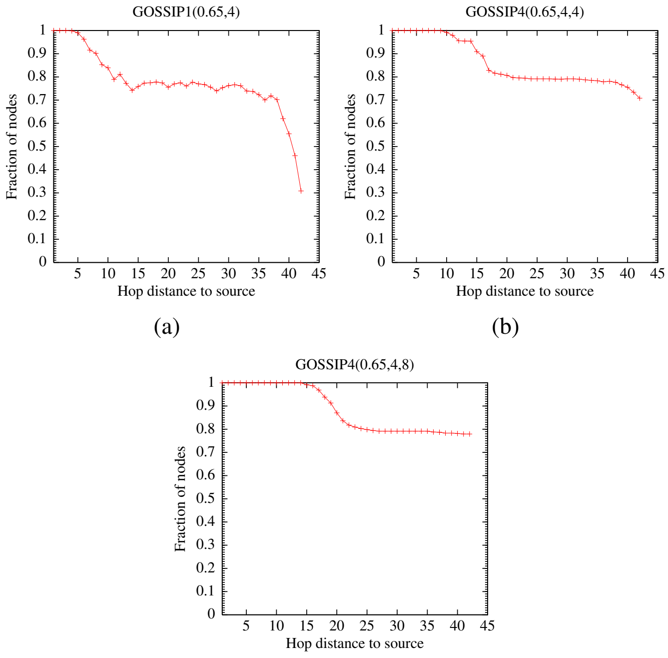
A node proactively tries to maintain complete routing tables for all nodes in its zone. Initially, a node discovers who its neighbors are and then broadcasts its neighbors to its zone (by using flooding up to hop count $\rho$). Then each time it discovers a change (i.e., that it has lost or gained a neighbor), it broadcasts this fact as well. This procedure ensures that a node has a very accurate picture of its zone.

If a source wants to send to a destination in its zone, it simply routes the message directly there, since it knows the route. Otherwise, it sends a route request query to the peripheral nodes in its zone. If the destination is in a peripheral node's zone, the peripheral node replies to the query originator. Otherwise, it forwards the query to its peripheral nodes, which in turn forwards it, and so on.

In the context of ZRP, there are two advantages of maintaining a zone. First, if a node is in the zone, flooding is unnecessary; a message can be sent directly to the intended recipient, saving much control traffic. This brings about a significant improvement in overall performance if a substantial fraction of nodes are in the zone (which is likely to be true in a small network, but far less likely in a large one). Second, if we want to send a message outside the zone, we can multicast to the boundary of the zone (or a subset of the nodes on the boundary), which can be a significant saving over flooding. However, there is a tradeoff in choosing the size of the zone: a bigger zone benefits more from the two advantages, but also results in more maintenance overhead. In general, the optimal zone size will depend on factors like mobility and frequency of route requests.

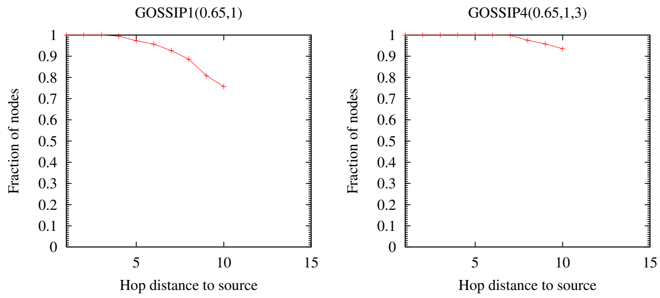
The idea of zones can be used in gossiping as well. Here there is a third advantage: if a node in the zone receives a gossip message, then it can send it directly to any node in the zone. This means that it would suffice for a gossiping protocol to get the message to a node in the intended recipient's zone. How much of an advantage is this? In large networks, the advantage is quite minimal. As we have observed, gossiping is essentially bimodal: for typical gossip probabilities, either hardly any nodes get the message or most of them do. Zones have a relatively small effect in either case. Thus, zones help only in the relatively few executions that exhibit "intermediate" behavior.

Let GOSSIP4($p, k, k'$) be just like GOSSIP1($p, k$), except that each node has a zone of radius $k'$. Comparing Figure 7(b) to Figure 7(a), we see using a zone radius of 4 with gossiping probability .65 in the random network with average degree 8 improves performance by only a few percent over most of the distances. However, it does ameliorate the back-propagation effect. As shown in Figure 7(c), increasing the zone radius to 8 does not significantly improve the limiting performance, but it has an even more beneficial effect on the back-propagation problem.




(a) (b) (c)

Fig. 7. Gossiping with zones on a random network of average degree 8.




(a) (b)

Fig. 8. Gossiping with zones on a 100 node random network.

The situation is much different for smaller networks. Here zones can have a significant impact. For example, if we use gossip probability .65 in a random network with 100 nodes and average degree 13, the network is too small for the bimodal effect to show up. However, the back-propagation problem is significant. As Figure 8 shows, for the small random network of 100 nodes, if we use GOSSIP1(0.65,1), then only 76% of nodes at distance 10 get the message. However, if we have a zone of radius 3 (GOSSIP4(0.65,1,3)), then 96% of nodes at distance 10 get the message.

## V. Incorporating Gossiping in AODV

How much does gossiping really help in practice? That depends, of course, on issues like the network topology, mobility, and how frequently messages are generated. We believe that in larger networks with high mobility many of the optimizations discussed in the literature will be much less effective. (We discuss this point in more detail below in the context of AODV.) In this case, flooding will occur more frequently, so gossiping will be particularly advantageous. However, as our results show, gossiping can provide significant advantages even in small networks.

To test the impact of gossiping, we considered AODV, one of the best-studied ad hoc routing protocols in the literature. We compared pure AODV to a variant of AODV that uses gossiping instead of flooding whenever AODV would use flooding. We do not have the resources to simulate the protocols in large networks. However, our results do verify the intuition that, with high mobility (when flooding will be needed more often in pure AODV), gossiping can provide a significant advantage.

### *A. A brief overview of AODV*

Using AODV, the first time a node $u$ requests a route to node $v$, it uses an *expanding-ring search* to find the route. That is, it first tries to find the route in a zone of small radius, by flooding. It then tries to find the route in zones of larger and larger radius. If all these attempts fail, it resorts to flooding the message through the whole network. The exact choice of zone radii to try is a parameter of AODV. Typically, not too many radii are considered before resorting to flooding throughout the network.

AODV also maintains a routing table where it stores the route after it has been found. If AODV running at node $u$ gets any packet with source $u$ and destination $v$, the route in the routing table will be tried. If any node $w$ on the route from $u$ to $v$ detects that the link to the next hop is down, then $w$ generates a route error (RERR) message, which is propagated back to $u$. When $u$ receives the RERR message, it deletes the route to $v$ from its routing table.

### *B. GOSSIP3 in AODV*

We added gossiping to AODV in a particularly simple way. If the expanding-ring search with a smaller radius fails, rather than flooding to the whole network, we use GOSSIP3(.65,1). (We used the parameters .65 and 1 since they gave good performance in the particular scenarios we considered.) The timeout period of GOSSIP3 should be big enough to allow neighboring nodes to gossip. The $NODE\_TRAVERSAL\_TIME$ parameter of AODV is a conservative estimate of the average one hop traversal time for packets that includes queueing delays, interrupt processing times and transfer times. In our experiments, we set the timeout interval to be $i * NODE\_TRAVERSAL\_TIME$ where $i$ is a small integer ($i = 5$ in our reported results). Note that we do not use GOSSIP3 in the expanding-ring search with a smaller radius. Because of the back-propagation effects, flooding is actually more efficient than gossiping for a zone with a small radius. We call the variant of AODV that uses GOSSIP3 AODV+G.

### *C. Simulation model and performance results*

Our simulation is done in the ns-2 [19] simulator. This is also the simulator the literature uses to evaluate AODV. We use the AODV implementation in ns-2 downloaded from one of the author's web site, using IEEE 802.11 as the MAC layer protocol. The radio model simulates Lucent's WaveLAN [21] with a nominal bit rate of 2Mb/sec and a nominal range of 250 meters. The radio propagation model is the two-ray ground model [20].

Our application traffic is CBR (constant bit rate). The source-destination pairs (connections) are chosen randomly. The application packets are all 512 bytes. We assumed a sending rate of 2 packets/second and 30 connections.

For mobility, we use the *random waypoint* model [3] in a rectangular field. The simulation scenarios are as follows: 150 nodes are randomly placed in a grid of $3300m \times 600m$; there are 30 connections, each generating 2 packet/sec; simulation time is 525 seconds; each node moves with a randomly chosen speed (uniformly chosen from 0-20 m/sec), then pauses for $\tau$ seconds after reaching a randomly set destination. We vary the pause time to simulate different mobility scenarios. Each data point represents an average of five runs using the identical traffic moel, but with different randomly generated mobility scenarios. To preserve fairness, identical mobility and traffic scenarios are used for both AODV and AODV+G.

We used the same configuration parameters for AODV as those used in [5]. Of particular interest to us are the expanding-ring search parameters. In the ns-2 implementation of AODV, first a zone radius of 5 hops is tried; if no route is found, network-wide flooding is used.

We study the performance of the following four metrics, of which the first three were also studied in [5]:

- The *packet delivery fraction* represents the ratio of the data packets delivered to the destination to those generated by the CBR sources.
- The *average end-to-end delay* of data packets includes all possible delays caused by buffering during routing dis-

covery, queuing at the interface queue, retransmission at the MAC layer, propagation, and transfer time.

- The *normalized routing load* represents the number of routing packets transmitted per data packet delivered at the destination. Each hop-wise packet transmission is counted as one transmission.
- The *route length ratio* compares the shortest route length found to the actual shortest route length.

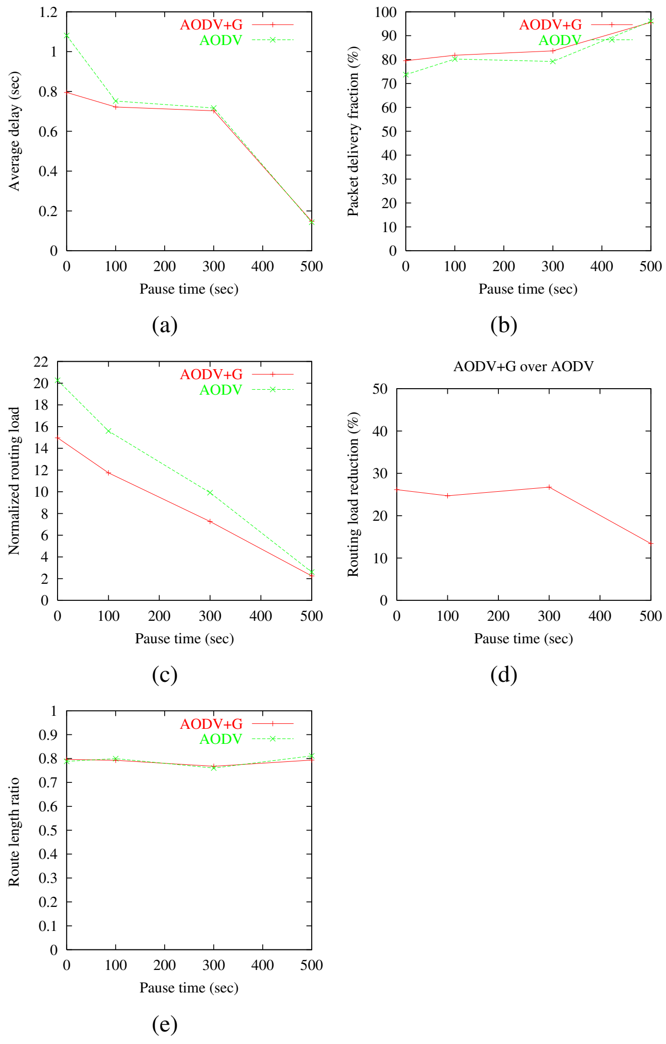



Fig. 9. AODV+G vs. AODV.

From Figure 9(a) and 9(b), we see that AODV+G delivers better network performance than AODV in terms of end-to-end delay and packet delivery fraction. The performance improvements correlate with the amount of routing load reduced. This is not surprising, since routing load increases with mobility and constitutes a significant part of the network load (as can be seen from Figure 9(c)). At pause time 0, AODV+G reduces average end-to-end delay by 36% and increases throughput by 8%. From Figure 9(c) and 9(d), we see that AODV+G reduces the routing load; the reduction is from 14% to 27% in terms of normalized routing load.

Finally, we consider route lengths. Note that neither gossiping nor flooding (as used by AODV) will necessarily find the shortest route. For example, suppose that $(u_0, u_1, u_2, u_3)$ is the shortest path from $u_0$ to $u_3$, but that there is another path $(u_0, v_1, v_2, u_2, u_3)$. It is possible that after $u_0$ broadcasts a route request, $u_2$ will receive it along the path from $v_2$ before receiving it from $u_1$. Since, in AODV, $u_2$ would save in its routing table information from only the first route request to arrive, AODV will not necessarily discover the shortest route. For similar reasons, with gossiping, we may not always discover the shortest routes. Nevertheless, a priori, it might seem that flooding will find shorter routes than gossiping. However, our results show that this is not particularly the case. We considered the ratio of the shortest route found by AODV to the actual shortest route, and similarly for AODV+G. Figure 9(e) shows that the routing length ratio for AODV+G and AODV is almost the same (and, indeed, is sometimes marginally better for AODV+G).

These simulations were carried out in a network with 150 nodes. In such a small network, even if route-destination pairs are chosen at random, a great many pairs will be within 5 hops of each other, and will thus be discovered by the expanding-ring search. Indeed, in our simulation, roughly 30%-40% of the routes discovered had a length of less than or equal to 5. Thus, as many as 40% of the routes are discovered by the expanding-ring search. We expect that things will be quite different in a larger network. Of course, that depends in part on the nature of route requests and the choice of parameters for the expanding-ring search. While it is possible that many requests will be local, there are applications for which this seems unlikely. Certainly if route-destination pairs are chosen at random, then expanding-ring search is unlikely to be effective for almost any choice of parameter settings. A great many source-destination pairs are likely to be far apart, so no expanding-ring search is likely to find them efficiently. Additionally, expanding-ring search may add a great deal of routing traffic and route discovery latency. By way of contrast, gossiping continues to perform well in large networks. Thus, we predict that the relative advantage of AODV+G over pure AODV will increase as the network gets larger. The graphs here will underestimate the performance improvement.

## VI. Concluding Remarks

Despite the various optimizations, with flooding-based routing, many routing messages are propagated unnecessarily. We show that gossiping can reduce control traffic up to 35% when compared to flooding. Our protocol is simple and easy to incorporate into existing protocols. When we add gossiping to AODV, simulations show significant performance improvements in all the performance metrics, even in networks as small as 150 nodes. As discussed in the Section V, we expect this performance improvement to become even more significant in larger networks.

We have also experimented with adding gossiping to ZRP, by using gossiping to send the route request to some peripheral nodes rather than to all peripheral nodes. Again, our results show significant improvement in all performance metrics. It seems likely that gossiping can be usefully added to a number of other ad hoc routing protocols.

Gossiping has a number of advantages over other approaches considered in the literature. For one thing, unlike many heuristics considered in the literature, we believe that we have a very good understanding of how gossiping will perform in large networks. This understanding is supported both by experimental evidence and analytical results. While there are fundamental limits to the amount of nonlocal traffic that can be sent in large networks, due to problems of scaling [8], [14], gossiping should still be useful in large networks when nonlocal messages need to be sent. It is far less clear how well other optimizations considered in the literature will perform in large networks. Moreover, as our simulation with AODV has shown, gossiping can provide significant advantages even in small networks. Experience in other contexts has shown that gossiping is also quite robust and able to tolerate faults; we expect that this will be the case in ad hoc routing as well. Finally, by changing the gossip parameters, it is easy to tune gossiping to a particular network and particular performance considerations. All this suggests that gossiping can be a very useful adjunct to the arsenal of techniques in mobile computing. Of course, work needs to be done in finding good techniques to learn the appropriate gossip parameters. We have exprimented on adjusting the gossiping probability of each node according to the success/failure of route requests; it is increased if the route request failure probability is high, and decreased if the route request failure probability is close to 0. To propagate the appropriate probability throughout the network, it can be put into the route request packet. Each intermediate node receiving the packet will gossip with the probability carried in the route request packet. Our preliminary experiments have shown that this approach does produce good results, although we have not had enough experience to determine the best way of making these adjustments to the gossip probability; we leave this for future work.

*Acknowledgments*

We would like to thank Jon Kleinberg for suggesting the relevance of percolation theory, Harry Kesten for explaining the relevant results of percolation theory, and Alan Demers for many useful comments.

## References

[1] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proc. Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 76–84, 1998.

[2] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, October 1998.

[4] R. Chandra, V. Ramasubramanian, and K. Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In *Proc. 21st International Conference on Distributed Computing Systems (ICDCS)*, pages 275–283, 2001.

[5] S.R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pages 3–12, March 2000.

[6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 1–12, 1987.

[7] G. Grimmett. *Percolation*. Springer-Verlag, New York,NY, 1989.

[8] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, 2000.

[9] Z. Haas and M. Pearlman. The performance of query control schemes for the zone routing protocol. In *Proc. ACM SIGCOMM*, pages 167–177, August 1998.

[10] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 174–185, 1999.

[11] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic Publishers, 1996.

[12] B. Karp and H. T. Kung. Greedy perimeter stateless routing (GPSR) for wireless networks. In *Proc. Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[13] Y. B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proc. Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 66–75, 1998.

[14] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 61–69, 2001.

[15] R. Meester and R. Roy. *Continuum percolation*. Cambridge University, 1996.

[16] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proc. fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 151–162, 1999.

[17] V. Park and M.S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. INFOCOM*, pages 1405–1413, April 1997.

[18] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[19] VINT Project. The UCB/LBNL/VINT network simulator-ns (Version 2). http://www.isi.edu/nsnam/ns.

[20] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[21] Bruce Tuch. Development of WaveLAN, an ISM band wireless LAN. *AT&T Technical Journal*, 72(4):27–33, 1993.

[22] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Duke Technical Report CS-2000-06, July 2000.