An Efficient Flooding Algorithm for Mobile Ad-hoc Networks

Jesus Arango
Computer Science
University of Arizona
Tucson, Arizona, USA
jarango@cs.arizona.edu

Mikael Degermark
Computer Science
University of Arizona
Tucson, Arizona, USA
micke@cs.arizona.edu

Alon Efrat
Computer Science
University of Arizona
Tucson, Arizona, USA
alon@cs.arizona.edu

Stephen Pink
Computer Science
University of Arizona
Tucson, Arizona, USA
steve@cs.arizona.edu

***Abstract*—This paper presents Geoflood, a bandwidth-efficient flooding algorithm intended for use in wireless ad-hoc networks. Flooding algorithms solve the problem of delivering a message to all nodes in a network. With pure flooding, each node broadcasts the message once in order to ensure delivery to all nodes in the network. Geoflood refrains from broadcasting when a message has been received from "many" directions. The new algorithm is simple to implement and does not require any additional protocol messages.**

Simulation results show a 69% reduction in bandwidth overhead for the highest simulated node density, which is just 24 nodes per in-range unit (48,087m^2). Geoflood reduces bandwidth overhead in networks with node densities as small as 3 nodes per in-range unit. Geoflood performs even better under high node mobility. All this while still maintaining optimal coverage and low latencies.

Keywords—geoflood; flooding; simulation; mobile; ad-hoc; networks; routing; geocast

I. INTRODUCTION

An ad-hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of a fixed infrastructure or centralized administration. Due to the limited propagation range of each mobile node, it may be necessary for one mobile host to enlist the aid of others in forwarding a packet to its destination.

Numerous routing protocols for ad-hoc networks have been proposed. Some relevant routing protocols are reviewed in [1]. Most ad-hoc routing protocols are categorized as either table-driven or on-demand.

Table-driven routing protocols such as DSDV [6] maintain routing information to every destination. Nodes respond to changes in network topology by sending updates that maintain a network view that is both consistent and up-to-date. DSDV is based on the classical Bellman-Ford routing algorithm.

On-demand routing protocols such as AODV [2] and DSR [7] use flooding algorithms for route discovery to construct on-demand routes. Flooding algorithms solve the problem of delivering a message to all the nodes in the network. Pure flooding requires each node to forward the message once. Several optimizations have been proposed to lessen the overhead of flooding during route discovery.

Flooding is particularly expensive for wireless networks, where bandwidth, battery and computational resources are often scarce. Flooding is also used in multicasting, geocasting, location discovery, sensor networks, etc.

An interesting use of flooding protocols is geocasting. The purpose of geocasting is to deliver messages to specific geographical regions defined by the user or application. Geocasting is a special case of multicasting, where the group members are confined to a geographical region. Geocast messages are delivered using flooding by defining what is called a forwarding zone. As shown in Figure 1 (taken from [3]), the forwarding zone includes the source and the geocast region. Flooding is limited to the forwarding zone. Nodes located outside the forwarding zone will drop the messages without further processing. Messages contain header fields that describe the location and geometry of the target region and the forwarding zone.

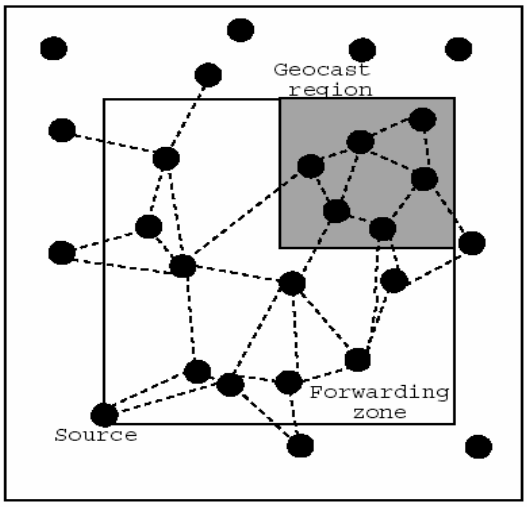

Figure 1: Forwarding region used in geocast flooding

This paper presents a new and efficient flooding protocol for mobile ad-hoc and sensor networks which

minimizes flooding overhead by significantly reducing the overall number of messages required to reach all nodes.

The algorithm assumes that each node can discern its own location, but it does not require each node to know the location of its neighbors. This is an important distinction since learning the location of other nodes is usually done by means of a "hello" protocol, which adds additional protocol messages. One might argue that protocols such as AODV already require a hello protocol. However, this is not necessarily true for other applications such as emerging geocasting [3][5] technologies.

Today, nodes can easily obtain their location through already popular GPS devices. It will become evident later that only a coarse knowledge of the location is required. Therefore, the algorithm can also use known ad-hoc location discovery methods which are less accurate than GPS but can be used indoors [12][13][14][15][16].

Not all nodes are strictly required to know their location, as our algorithm can be seen as an optimization for pure flooding where savings in bandwidth overhead increase with the number of location-aware nodes.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 describes our optimized flooding algorithm. Section 4 presents the performance evaluation and simulation results of our algorithm. Section 5 includes some additional discussions and future work. Finally, Section 6 provides some concluding remarks.

II. Related Work

V. Paruchuri, *et. al.* [8] proposed an efficient flooding algorithm based on the covering problem, which states that the minimum number of circles required to cover a 2-dimensional space is obtained by overlaying a hexagonal lattice arrangement on the plane and circumscribing a circle around each hexagon. For flooding in ad-hoc networks, one would additionally require that the center of each circle lie on the circumference of all adjacent circles (one node within range of another). This requirement can be met with an alternate arrangement where one circle is centered on each vertex of the hexagonal lattice. The algorithm works by having the message forwarded only by those nodes that share a hexagon edge with the local sender.

The authors claim to use up to 65% to 80% fewer messages than pure flooding. Their highest simulated node density was 2.22 times greater than the highest node density we simulated, on which we achieved a 69% improvement.

Paruchuri's optimized flooding is the best solution we found in the literature, so we think it is worth mentioning some differences with our protocol. Their more sophisticated approach also makes it more complex. Their algorithm requires a more precise positioning system. Our algorithm also differs in that it does not require a hello protocol. The authors briefly describe a variant of their algorithm that does not require hello messages, but did not implement this variant, nor did they elaborate on it or provide results.

Z. Hass *et. al.* introduced GOSSIP – essentially, tossing a coin to decide whether or not to forward the message – to reduce the overhead of routing protocols. Gossiping exhibits bimodal behavior in sufficiently large networks: in some executions the gossip dies out quickly and hardly any nodes get the message, in the remaining executions most of the nodes get the message. This simple gossiping protocol uses up to 35% fewer messages. Compared to GOSSIP, our approach achieves less overhead and better coverage (the fraction of nodes that receive the message).

J. Cartigny, *et. al.* [10] proposed several stochastic algorithms where nodes forward messages with a certain probability. These probabilities are calculated differently for each algorithm presented in the paper. The choices go from using a constant probability (like GOSSIP) to calculating the probability as a function of local node density, distance between sender and receiver and fraction of neighbors that have received the message, or a combination of these. These algorithms provide a significant reduction of forwarded messages at the cost of less coverage/reliability. As in GOSSIP, our approach yields less overhead without compromising coverage.

III. GeoFlood Algorithm

In ad-hoc networks, pure flooding works as follows. When a node receives a message, the node first determines whether it is the first reception of the message. If it is the first reception, the node will forward the message, otherwise it drops the message. Each message is stamped by the originator with a unique sequence number. Nodes can detect duplicates by keeping a record of *(source, sequence)* pairs for previously received messages.

We improve pure flooding by having each node wait a "small" period of time before forwarding on the first reception of the message, and abstaining from forwarding when it receives the same message from "all directions".

Each message contains a location field. As a message is forwarded, each intermediate node updates the location field with its own position. Each node defines a Cartesian plane with its own location as the origin. A node will abstain from forwarding only when it has received the message from all four quadrants (*NE, NW, SE, and SW*).

Thus the algorithm works as follows: If the received message has been forwarded earlier by the local node, the message is dropped. If this is the first reception of the message, the quadrant from which the message arrived is recorded, and a packet holding time t is chosen. The message is temporarily put on hold until

either the message is received from all four quadrants or *t* time has passed. If the message arrives from all four quadrants before time *t*, then the message is dropped, otherwise it is forwarded and the (*source, sequence*) pair is stored in the forwarding cache to filter future duplicates.

Location knowledge is not strictly required on all nodes. Geoflood is just an optimization for pure flooding where location-aware nodes can help reduce the bandwidth overhead. A node that does not know its location will do no waiting and will immediately forward messages with the location field empty. A location-aware node that receives a message with an empty location field will not assign the message to a quadrant.

An important part of the algorithm is the selection of packet holding time. Nodes furthest away from the local sender should select the smallest packet holding times. These are the nodes located near the perimeter of the sender's transmission range. Holding times increase as the distance to the sender decreases, with those nodes closest to the sender waiting the longest. A random component was also considered to avoid contention that could arise between nodes located at the same distance from the sender. As will be discussed later, simulation results indicate this random component does not have the intended effect.

This time selection serves two purposes. As shown in Figure 2, outer nodes cover more new network space than inner nodes, thus smaller waiting times for outer nodes help messages propagate faster and keep latencies down for "far-away" nodes. Moreover, if enough outer nodes are able to transmit quicker, inner nodes that don't cover much additional space will desist from forwarding if they receive the message from outer nodes on all four quadrants.

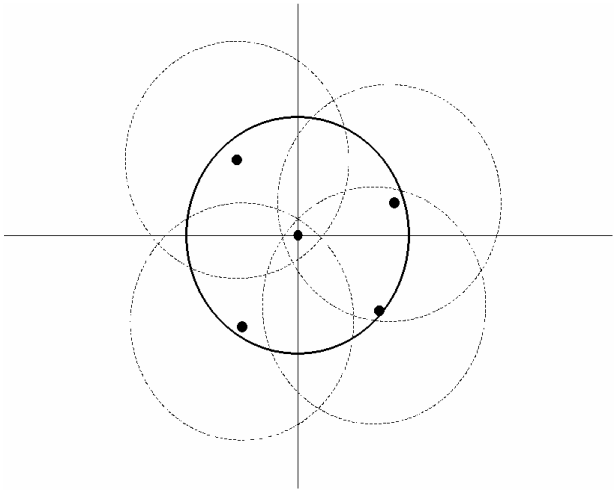

Figure 2: network space covered by outer nodes

We propose a simple linear function to implement this timing strategy. Let $\boldsymbol{r}$ be the range of the network. Let $\boldsymbol{m_{hold}}$ be the maximum holding time. Let $\boldsymbol{d}$ be the distance between the local node and the sender. Let $\boldsymbol{h}$ be the function that computes the packet hold time. The function $\boldsymbol{h}$ is defined as follows:

$$h(d) = m_{hold} - \frac{d \cdot m_{hold}}{r}$$

Let $\boldsymbol{m_{off}}$ be the maximum random offset. A random component may be added to the packet hold time by means of the following function.

$$t(d) = rand(\max(h(d) - m_{off}, 0), h(d) + m_{off})$$

IV. PERFORMANCE EVALUATION

Performance evaluation was carried out by simulating geoflood and pure flooding under identical network scenarios and comparing the results.

We implemented optimized and pure flooding in the ns network simulator [11]. Ns is a discrete event simulator targeted at networking research. Ns provides extensive support for wireless networks.

Three metrics were defined to evaluate and compare the performance of both protocols: (1) *Overhead*: the ratio between the number of times a message is forwarded and the number of nodes in the network. (2) *Coverage*: the ratio between the number of distinct nodes that receive the flooded message and the number of nodes in the network. (3) *Latency*: the average time it takes a node to receive the message from the time the message was originated.

For each simulation of *n* nodes, *n* flooding operations were conducted, with each node performing one flooding operation. The average overhead, average coverage and average latency is calculated over the *n* flooding operations.

The simulations were conducted with the following fixed conditions: Nodes are uniformly distributed in a network space of dimensions 1000x1000 meters. The default ns node configuration was used, which works like a 914MHz Lucent WaveLAN DSSS, namely, a bandwidth of 2Mbps and a mean transmission range of 250 meters. We used the size of a typical AODV RREQ message as the packet size for the simulations. AODV RREQ messages have a payload of 24 bytes.

For geoflood, the simulations were conducted with a maximum holding time ($\boldsymbol{m_{hold}}$) of 350 milliseconds and a maximum random offset ($\boldsymbol{m_{off}}$) of zero milliseconds.

Simulations were conducted for multiple node densities. Node densities are expressed as the number of nodes per *in-range* unit. An in-range unit is the largest circle such that any node inside the circle can directly communicate with any other node inside the circle. For a transmission range of *250* meters the in-range unit is *49,087* square meters. Several node distributions were

simulated for each (*node density*, *algorithm*) pair and the results were averaged. The results are presented for each metric by plotting the metric vs. the density for both geoflood and pure flooding.

Figure 3 shows the results for the overhead metric. Geoflood performs much better than pure flooding as the node density increases, achieving a 69% reduction for the largest density we simulated (24.54). From an arbitrary node's perspective, the higher the density the higher the likelihood that the message will be received from all four quadrants.

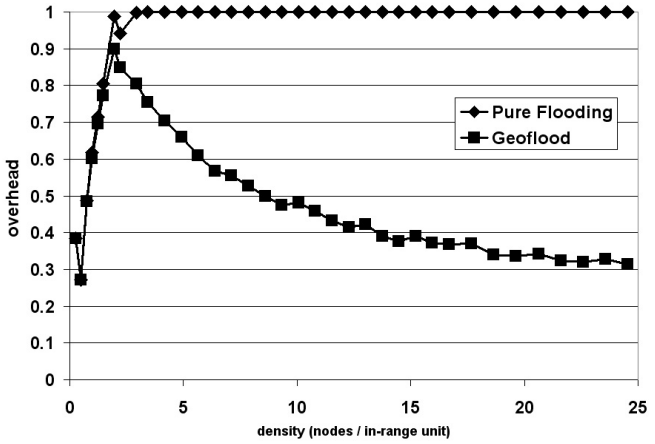

Figure 3: Overhead vs. Density

Geoflood starts reducing bandwidth overhead as soon as the node density is such that the ad-hoc network is fully connected. Under a uniform node distribution, only minor densities are enough to produce a fully connected network. A node density of 3 nodes per in-range unit is enough to provide a fully connected ad-hoc network.

Full connectivity in Figure 3 occurs where the overhead peaks. Note that the overhead peaks at about the same density for both algorithms. This corresponds to a node density of 2.94. The reduced overhead for densities less than 2.94 are a result of how the overhead is measured. Recall that overhead is measured as the fraction of the nodes that broadcast the message. When the network is not fully connected, not all nodes can receive the message, hence not all nodes will forward the message. This is true for any flooding algorithm. For node densities less than 2.94, the overhead acts as a measurement of connectivity.

As Figure 4 shows, geoflood exhibits excellent coverage. The algorithm provides coverage that is just as good as the coverage for pure flooding. Both pure flooding and geoflood exhibit perfect coverage once the node density is such that the ad-hoc network is fully connected.

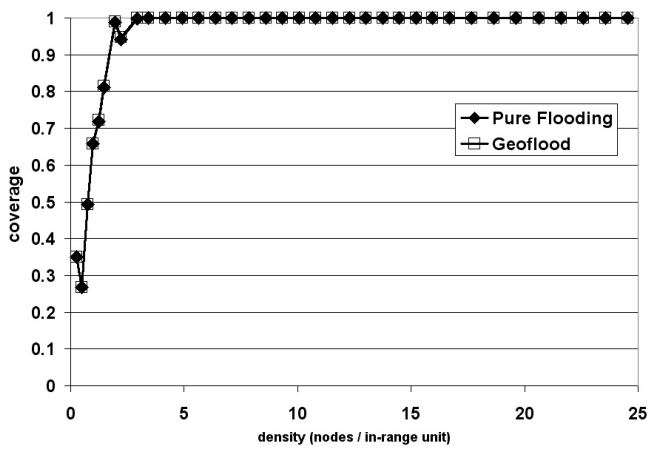

Figure 4: Coverage vs. Density

This is also consistent with our expectations: If a node does not receive a message from all four quadrants it will eventually forward the message, thus covering the same network area as any node in pure flooding would. On the other hand, when a node receives the message from all four quadrants, the area that is covered by its range but not covered by the range of any of the four senders is either null or very small. Under full connectivity it is therefore very likely that any other node in the receiver's range will either receive the message from one of the four senders or perhaps another node. Figure 5 illustrates a worst-case scenario where the area covered by the receiver but not covered by one of the four quadrant senders is the largest. Note that even in this worst-case scenario the area not covered is still relatively small. Moreover, that area would likely be covered by other nodes when reasonable densities are considered.

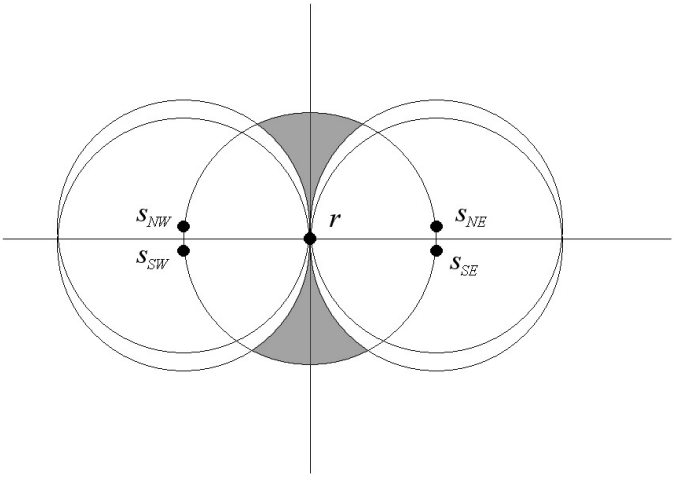

Figure 5: Worst case placement of local senders from all four quadrants.

Figure 6 shows how the average delivery latency decreases rapidly as the node density increases. For the highest simulated node density of 24 nodes/in-range unit, the average latency of our algorithm was 27

milliseconds, just 10 milliseconds more than pure flooding.

Most of the decrease in latency occurs within a reasonable/realistic node density interval. For example, at a node density of 2.20, close to the density threshold for fully connected networks, the average latency is 156 milliseconds. At a node density of 10.79, the latency is already down to 37 milliseconds, just 20 milliseconds more than pure flooding, and just 10 seconds more than the highest simulated density. This is a 76% decrease in latency between node densities 2.20 and 10.79.

The decrease in latency as the density increases is easy explained, and is a result of the packet hold time function used. Neighbor nodes that are furthest away from the transmitting node generate the smallest random holding times. This allows these far reaching nodes to transmit quickly and reach greater distances in a short time.

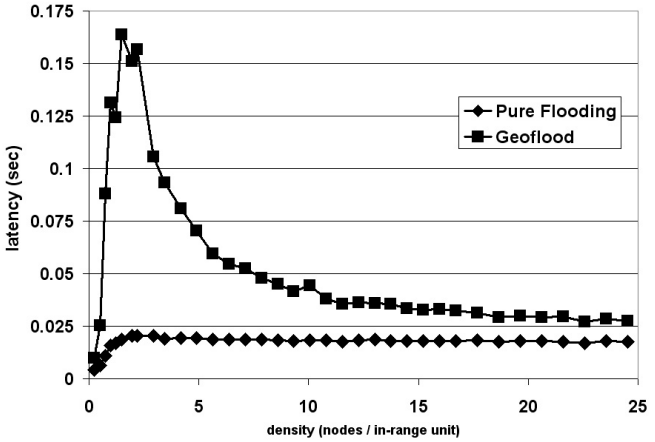

Figure 6: Latency vs. Density

A. Packet Holding Times

Separate simulations were conducted to determine the effect of the packet holding time and random offset on the performance of the algorithm. Simulations were carried out for packet holding times between 0.0 and 1.0 seconds with a step of 10 milliseconds, and a random offset between 0 and 50 milliseconds with a step of 2.5 milliseconds. All simulations were conducted on a fixed node density of 6.38 nodes per in-range unit. The results of the simulations are used to analyze the effect of these times on each performance metric.

Figure 7 shows a 3D plot of the bandwidth overhead as a function of packet hold time and random offset. Several observations can be made by analyzing the figure. First, the surface flattens rapidly as the packet hold time increases. The contours drawn at the base of the graph indicate most of the overhead reduction occurs at small packet hold times. The overhead drops below 80% at 10ms, below 70% at 90ms, below 65% at 130 ms, below 60% at 300 ms, and below 57% at 1 second of packet holding time. It is then reasonable to say that after 300ms of packet hold time there is no significant savings in bandwidth overhead.

It can also be observed that there is no clear benefit in introducing a random offset. A careful inspection of the figure and its contour for an overhead of 0.58 shows that the random offset is actually detrimental to overhead reduction. The simulation data indicates that the random offset produces a negative effect for packet hold times longer the 100 ms, with greater random offsets resulting in increased overhead. It should be noted that significant overhead reduction can still be achieved for packet hold times greater than 100 ms. For example, the overhead is reduced an additional 10% between 100 ms and 350 ms of packet hold time.

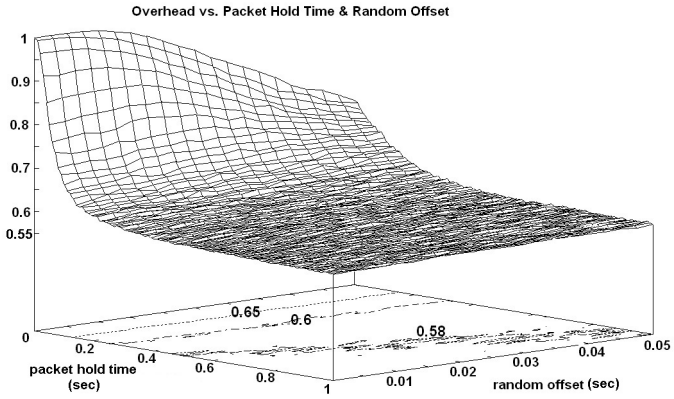

Figure 7: Bandwidth overhead as a function of packet hold time and random offset

The opposite effect occurs for packet hold times smaller than 100 ms. Figure 8 shows a zoomed view of this region. The surface and contours show how longer random offsets result in less overhead. This does not mean, however, that the random offset is fulfilling its original purpose. Namely, the random offset was supposed to avoid contention among nodes located at the same distance from the transmitting node. What we are seeing instead is the random offset compensating for an otherwise small packet hold time. In other words, a random holding time is better than no holding time.

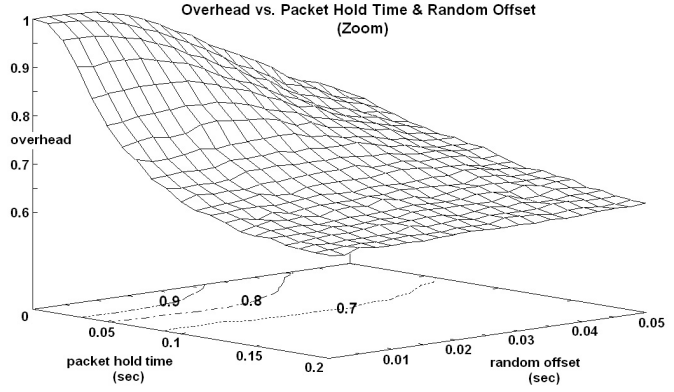

Figure 8: A Zoomed view of Figure 7

The average latency for delivery of a flooded packet increases almost linearly with the packet hold time. This is shown in Figure 9. Latency-wise, good judgment is at a premium when selecting the packet holding time. There is a packet hold time interval where latency increases at a much faster rate than the decrease in overhead. It is not recommended to use packet hold times beyond 300 or 350 ms for the network configuration we have simulated.

The coverage was always 1.0 for all (*packet hold time, random offset*) pairs. For this reason it is pointless to include the corresponding figure in the paper.

The analysis we have conducted with respect to holding times is consistent across different network configurations. We conducted similar simulations for networks with different densities, areas and bandwidth and always obtained similar results.

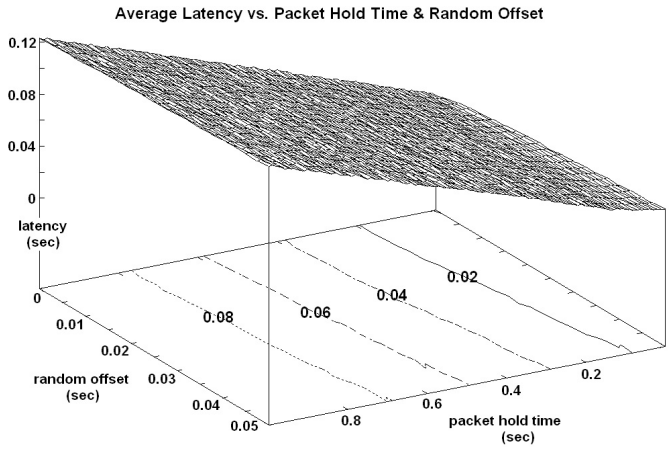

Figure 9: Latency as a function of packet hold time and random offset

B. Movement

To provide a complete study of our protocol we also conducted simulations in the presence of node mobility. Node mobility does not adversely affect the performance of our protocol. In fact, the results consistently indicate that our protocol actually performs better when nodes are moving.

Simulations were carried out using the following mobility model: each node picks a random destination using a uniform distribution. The node moves towards that destination at a constant speed between 0 and 20 m/s (72 km/h, 45 mi/h). Once the destination is reached, a new destination is selected and the process is repeated. All other conditions are the same as in the static simulations.

The overhead results for mobility are shown in Figure 10. The maximum node density simulated for mobility was 8.0 because simulation times became intolerable for larger densities. We have included the previous static results for convenience and easy comparison. The four lines represent both algorithms with and without mobility. The graph shows how geoflood performs consistently better under mobility. Note that mobility has no effect on pure flooding.

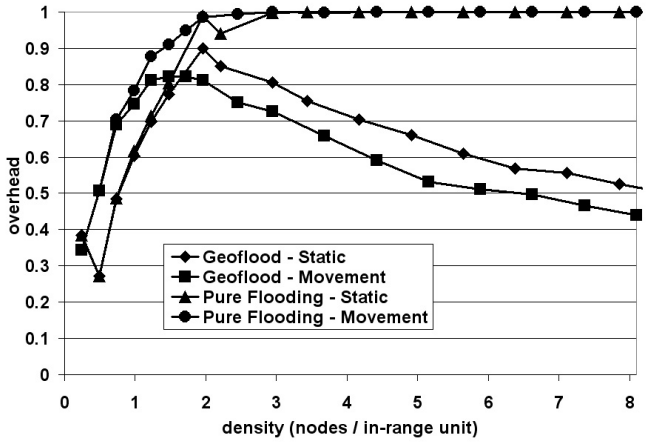

Figure 10: Comparison of overhead when considering node mobility

V. Discussion and Further Work

On-demand ad-hoc routing protocols such as AODV use flooding for route-discovery messages. If pure flooding is replaced with geoflood then not all nodes will forward the route discovery messages, and they will thus not be part of paths to the destination. The quality of the resulting paths is not clear. On one hand, paths will most likely have fewer hops because of our timing strategy where outer nodes retransmit first. On the other hand, these routes could be less stable as there is less overlap between the radio ranges of nodes along the path. The quality of the resulting routes needs to be verified. AODV is a natural protocol to target for such experiments.

The number of directions from which messages need to arrive before deciding not to forward will have an effect on the overhead and coverage metrics. Further studies need to be conducted to measure this effect. For example, the network space could be divided into three triangles instead of four quadrants.

VI. Conclusions

We have presented a scalable and bandwidth-efficient method for performing flooding in mobile ad-hoc and sensor networks.

The algorithm significantly reduces the bandwidth overhead under reasonable and realistic node densities. Experiments show that the algorithm provides the same coverage as pure flooding.

Because of its timer strategy, the algorithm is able to maintain the delivery latency under reasonable bounds while providing an efficient reduction in bandwidth overhead.

Our algorithm assumes that nodes know their own location. Only a coarse knowledge of the location is required. Therefore, the algorithm can use known ad-hoc location discovery methods which are less accurate than GPS but can be used indoors. Location knowledge is not strictly required on all nodes. Geoflood is just an optimization for pure flooding where location-aware nodes can help reduce the bandwidth overhead.

REFERENCES

[1] E. Royer and C.-K. Toh, *"A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks"*, IEEE Pers. Commun., vol. 6, no. 4, Apr. 1999, pp 46–55.

[2] C. Perkins, "*Ad hoc On Demand Distance Vector (AODV) Routing*", Internet draft, draft-ietf-manet-aodv-00.txt.

[3] X. Jiang, T. Camp, "*A Review of Geocasting Protocols for a Mobile Ad Hoc Network*", Proceedings of the Grace Hopper Celebration (GHC), 2002.

[4] Y. Ko, N. Vaidya, "*Flooding-Based Geocasting Protocols for Mobile Ad Hoc Networks*", MONET 7(6): 471-480 (2002).

[5] R. Jain, A. Puri, R. Sengupta, "*Geographical Routing using Partial Information for Wireless Ad Hoc Networks*" UCB/ERL Memo no. M99/69, December 1999. Also IEEE.

[6] C. Perkins, P. Bhagwat, "*Highly Dynamic Destination-Sequenced Distance-Vector (DSDV) routing for mobile computers*", In ACM SIGCOMM Symposium on Communication Architectures and Protocols, 1994.

[7] D. Johnson, D. Maltz, J. Broch, "*The dynamic source routing protocol for mobile ad hoc networks*", Internet Draft, March 1998.

[8] V. Paruchuri, A. Durresi, D. Dash, R. Jain, "*Optimal Flooding Protocol for Routing in Ad-hoc Networks* ", Submitted to IEEE Wireless Communications and Networking Conference (WCNC 2003), New Orleans, Louisiana, March 16-23, 2003.

[9] Z. Hass, J. Halpern, L. Li, "Gossip-Based Ad Hoc Routing", In IEEE INFOCOM, June 2002.

[10] J. Cartigny, D. Simplot and J. Carle, "*Stochastic flooding broadcast protocols in mobile wireless networks*", Tech. Report LIFL Univ. Lille 1 2002-03. may 2002.

[11] The VINT project, "*The ns manual*", A collaboration between researchers at UC.

[12] D. Niculescu and B. Nath, "*Ad Hoc Positioning System (APS) using AoA*", In Proceedings of INFOCOM 2003, San Francisco, CA.

[13] P. Bahl and V.N. Padmanabhan, "*Radar: An In-Building RF-Based User Location and Tracking System*" Proc. IEEE Infocom, IEEE Press, Piscataway, N.J., 2000, pp. 775–784.

[14] C. Savarese, J. Rabaey, and J. Beutel. "*Locationing in distributed ad-hoc wireless sensor networks*". In IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), pages 2037--2040, Salt Lake City, UT, May 2001.

[15] C. Savarese, J. Rabay, K. Langendoen, "*Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks*", In USENIX Technical Annual Conference, Monterey, CA, June 2002.

[16] J. Hightower, G. Boriello and R. Want, "*SpotON: An indoor 3D Location Sensing Technology Based on RF Signal Strength*", University of Washington CSE Technical Report #2000-02-02, February 2000.