

A Priority Forwarding Technique for Efficient and Fast Flooding in Wireless Ad Hoc Networks

Kihwan Kim Ying Cai Wallapak Tavanapong
Department of Computer Science
Iowa State University
Ames, IA 50011-1040
Email: {khkim, yingcai, tavanapo}@cs.iastate.edu

Abstract—In this paper, we propose a new technique, called *Priority Forwarding*, for efficient and fast flooding operations in wireless ad hoc networks. The new scheme is featured by *dynamic delay* and *priority checking*. The former feature allows a host to wait as long as possible to refrain from retransmission, minimizing retransmission overhead. The priority checking feature, on the other hand, allows a flooding packet to be propagated as quickly as possible, keeping flooding latency low. Unlike many location-aided flooding techniques, Priority Forwarding requires each host to know only the distance of its 1-hop neighbors, instead of their exact locations. Therefore, it has low implementation cost. For performance evaluation, we compare Priority Forwarding with some existing techniques using simulation. Our results show that under most scenarios, the new technique performs many times better in reducing packet retransmissions and flooding latency.

KEYWORDS: ad hoc networks, flooding, propagation, transmission coverage.

I. INTRODUCTION

In recent years, wireless ad hoc networks have received a great deal of research interest due to their simplicity and low cost of deployment. No pre-existing communication infrastructure is required. A node can communicate directly to nodes within its transmission range or to those outside its transmission range via wireless packet relays. Flooding, i.e., network-wide broadcasting, is an operation for sending a packet to all the other nodes in the network. In wireless ad hoc networks, flooding is necessary for service and resource discovery and is a building block for many unicast and multicast routing protocols (e.g., DSR [1], AODV [2], ZRP [3], LAR [4], just to name a few).

As a communication primitive, flooding has a significant impact on the overall performance of wireless ad hoc networks. Numerous flooding schemes have been proposed along with a number of metrics to evaluate their performance and implementation cost. For ease of exposition, we assume a static network and no packet collision, under which these techniques generally work best. To measure the performance of a flooding technique, a commonly used metric is *broadcast reachability* defined as

the percentage of reachable hosts that actually receive the broadcast packet. Reachable hosts are hosts in the network that can receive the broadcast packet through *simple flooding* [5][6], in which every host receiving a flooding packet is required to rebroadcast it once. A broadcast reachability below 100% means that some host does not receive the broadcast packet because an intermediate host between itself and the source decides not to retransmit the packet. Another metric is *retransmission overhead* defined as the percentage of receiving hosts (i.e., hosts receiving the broadcast packet) that actually retransmit the packet. The retransmission overhead of Simple Flooding is 100% since all hosts must transmit the packet once during a flooding operation. Low retransmission overhead is desirable to reduce unnecessary network traffic. The third metric is *broadcast latency* defined as the time taken by a flooding packet from a source to reach the last reachable host in the network. This metric has recently gained more attention since it is critical for time-constrained applications such as broadcasts of emergency messages and real-time multimedia applications. While these three metrics measure the performance of a flooding technique, its implementation cost is often measured by two factors. One factor is the number of hops that it requires each host to track its neighbors. In general, the more hops a host needs to track, the more *network control overhead* incur. Another factor is whether or not this technique relies on host location information, which may require positioning systems such as GPS [7].

An ideal flooding technique should simultaneously achieve three performance goals, i.e., 100% reachability, low transmission overhead, and low broadcast latency, with a low implementation cost. Existing techniques, however, fall short in achieving these goals. While some of them (e.g., [8], [9], [10], etc.) make a tradeoff between broadcast reachability and transmission overhead, many others (e.g., [11], [12], [13], [14], [15], [16], [17], etc.) either require excessive network control overhead, or may incur long broadcast latency, or both. Note that minimizing transmission overhead and minimizing broadcast latency are two seemingly conflicting goals. To keep retransmission

overhead low, some techniques (e.g., [18], [19], etc.) allow a host to keep track of redundant packets received over some time period. The value of this period is either randomly chosen between zero and some pre-configured value, or based on factors such as a host’s distance to where it receives the packet for the first time. At the end of this period, a host checks all redundant packets it receives and then determines whether or not to rebroadcast the packet. A longer period allows a host to collect more redundant packets, giving it a better chance to avoid retransmission of the packet. However, it tends to increase the broadcast latency.

In this paper, we propose a novel technique called *Priority Forwarding*, aiming at addressing the aforementioned problems. In our technique, upon receiving a packet, a host first checks its forwarding priority by finding out which one of its neighbors are reached by this packet. If none of these neighbors is closer than this host to the sender of the packet, this host forwards the packet immediately. Otherwise, it waits and dynamically adjusts its waiting time upon the receiving of duplicate packets. The advantages of our techniques are twofold. With the priority checking, a host can forward a packet immediately, minimizing the delay of flooding propagation. On the other hand, with the dynamic delay, a host can wait as long as possible to collect more duplicate packets, eliminating a large amount of unnecessary retransmissions. As our performance study indicates, with the proposed scheme, a broadcast is typically forwarded by the hosts close to the perimeter of a broadcast coverage. Therefore, it is highly scalable with respect to the network size and density. In addition, Priority Forwarding requires each host to know only its distance to its 1-hop neighbors. Thus, its implementation is lower than many existing techniques that require each host to track the locations of its neighbors within two or more hops.

The remainder of this paper is organized as follows. We present the concept of Priority Forwarding in Section 2, and introduce the protocol in Section 3. In Section 4, we describe the simulation model and examine the performance results. The concluding remarks are given in Section 5.

II. CONCEPT OF PRIORITY FORWARDING

In our technique, each host tracks its 1-hop neighbors through periodic heartbeat. In addition, we assume each host can determine its distance to each of its neighbors, which could be done by measuring the radio signal itself [23].

A. Dynamic Delay

When a host receives a new flooding packet, it can wait for a while. During the waiting time, it may collect duplicate packets forwarded by other hosts. If these packets have covered all its 1-hop neighbors, then this host does not have to forward the packet. The challenge is how to set a host’s waiting time. In this paper, we propose a *dynamic delay* approach. Suppose host X receives a new flooding packet, say, from S . X can initialize its waiting time W on this packet as $W = Max_Wait_Time \cdot \frac{R_S - dist(X,S)}{R_S}$, where R_S is the transmission radius of S and $dist(X,S)$ is the distance between X and S . If T ($T \leq W$) time units later, X receives a duplicate copy from Y , then X adjusts its waiting time on this packet to be $W = max(W, Max_Wait_Time \cdot \frac{R_Y - dist(X,Y)}{R_Y}) - T$, where R_Y is the transmission radius of Y . The basic idea is to allow a host to extend its waiting time as long as possible, but no more than Max_Wait_Time , to collect more duplicate retransmissions. Obviously, dynamically adjusting a host’s waiting time gives a host a better chance to avoid its forwarding obligation with the tradeoff of more computation.

As an example, consider Figure 1. When hosts A , B , and C receive a flooding packet from S , A ’s waiting time will be the shortest, B second, and C third. This is a good forwarding order at this moment because A is the one furthest to S , a broadcast from A will cover more new area than from B or C . However, after A forwards the packet, it would be better for C , instead of B , to forward next. Because B is very close to A , most of B ’s transmission area has already been covered by the broadcast of S and A . If we make C forward before B , B will not need to forward because the three duplicate packets it receives have already fully covered its transmission area. Apparently, our dynamic-delay approach reflects this need.

We note that using packet delay to preserve retransmission order was first explored in [18]. However, given a flooding packet, their approach sets a host’s waiting time only once, i.e., at the time when it first receives the packet. As we will see shortly in our performance study, such *static* delay setting retains a large amount of unnecessary retransmissions. Especially, when network density is not very high, its performance is almost the same as simple flooding. If we apply this scheme to the previous example, then B would have to forward after A does.

B. Priority Checking

Relying on packet delay to avoid packet retransmission may result in unnecessary propagation latency. For example, in Figure 1, after S sends its flooding packet, A

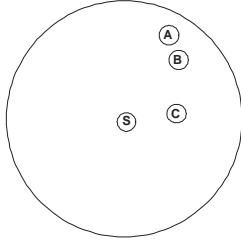


Fig. 1. Example

will wait $W = Max_Wait_Time \cdot \frac{R_S - dist(S,A)}{R_S}$ time units before forwarding the packet. Such waiting is unnecessary because A is the furthest one and should forward the packet immediately. After A forwards, C is the furthest one from A 's position. Again C should retransmit right away, instead of waiting another $W = Max_Wait_Time \cdot \frac{R_S - dist(S,C)}{R_S} - W_a$ time units, where W_a is A 's waiting time.

Such unnecessary forwarding delays can be avoided with the following *priority checking* mechanism. When a host broadcasts a flooding packet, it includes all of its 1-hop neighbors in the packet's header. Assume a host, say H , receives a flooding packet, say from S , whose transmission radius is R_S . Then H first finds out all of its 1-hop neighbors that are listed in the packet header. These hosts are common neighbors to both H and S . For each of such common neighbors, say N , H calculates $dist(N, S)$, the distance between N and S . If none of their distance to S is larger than H 's distance to S , then H has the highest priority to forward. In this case, H forwards the packet as soon as it acquires a clear communication channel.

When H receives a packet from S , it is possible that H can simply calculate the distance between its neighbors and S . Given a neighbor, say N , if $dist(N, S) \leq R_S$, then N should be in the transmission coverage of S . In reality, however, a wireless broadcast in most cases does not have a perfect coverage circle because of factors such as broadcast energy fading. Therefore, including neighbor list in a packet header makes our scheme more robust.

With our priority checking, a flooding packet can be propagated as soon as possible. For each broadcast, at least one broadcast will follow immediately, if there are still some hosts not covered. In the next section, we present a new flooding protocol, an integration of dynamic delay and priority checking, to achieve the following two seemingly conflicting goals:

- A host should wait as long as possible to collect more duplicate packets to avoid retransmission;
- A flooding packet should be forwarded as quick as possible to reduce flooding latency.

III. PROPOSED PROTOCOL

In Priority Forwarding, when a host sends a flooding packet, it includes its 1-hop neighbors in the packet's header. Each flooding packet is associated with a life thread. Upon receiving a flooding packet, a host spawns a new thread to handle the packet if this packet has not been received before. For this packet and all its duplicates received later, this thread does the following steps:

- 1) Check its neighbors one by one and mark those who are listed in the packet's header;
- 2) If all neighbors have been marked, then this thread terminates itself;
- 3) Otherwise, do priority checking:
 - For each of the newly-marked neighbors (i.e., those who receive this flooding packet first time), calculate its distance to the packet's sender;
 - If none of them is closer than the current host to the packet's sender, this thread forwards the packet and terminates itself;
- 4) Otherwise, set waiting time:
 - If this packet is first time received, it initializes a timer on this packet with a distance-based waiting time;
 - Otherwise, adjust the timer's waiting time, if possible, using the dynamic delay algorithm discussed early.

When the thread forwards the packet and terminates itself, it also destroys the packet's timer. If the duplicate packets collected by the host during its maximum waiting period cannot cover all of its 1-hop neighbors, the host forwards the packet and terminates its corresponding thread. Since a host has to rebroadcast a flooding packet unless all of its 1-hop neighbors can receive the packet from other hosts, our technique guaranteed 100% broadcast reachability. We also note that we exclude those neighbors who have received the flooding packet before and may have already forwarded the packet. By checking only newly-marked neighbors, we can ensure that after a broadcast is sent, at least one more broadcast will follow immediately if the flooding has not covered the entire network. Thus, the propagation delay can be minimized. On the other hand, dynamically adjusting a host's waiting time allows the host to wait as long as possible to collect duplicate packets, if this host is not the one furthest to a packet's sender. As we will see in performance study, this feature avoids a large amount of unnecessary retransmissions. Another major advantage of Priority Forwarding is its low control overhead. It requires each host to know only its distance to its 1-hop neighbors.

IV. PERFORMANCE STUDY

For the purpose of performance comparison, we have implemented detailed simulators for *Priority Forwarding*, *Simple Flooding*, and *Flooding with Self-Pruning* [11]. The last two techniques do not use forwarding delay. Their flooding latency is mainly caused by packet collision and channel competition. Since we are mainly interested in the reduction in unnecessary broadcast retransmission and their relative performance in flooding latency, we did not simulate the communication synchronization among the hosts, and assumed that a host could acquire a clear communication channel whenever it needed. Therefore, we are not able to determine the flooding latency caused by these two techniques. In order to compare both retransmission rate and flooding latency, we implemented another flooding technique, which was proposed in [18]. This scheme also makes a host wait upon receiving a new flooding packet, but it uses a fixed waiting time, which is set when the host receives a flooding packet at its first time. This mechanism is called *Distance-Based Defer Time* (DBDT). At the end of its waiting, the host checks the transmission coverage of each duplicate packet and if their concatenation can cover this host's entire transmission area, then it does not need to forward the packet. This mechanism is called *Angle-Based Scheme* (ABS). For simplicity, we will refer to this flooding technique, a combination of DBDT and ABS, as *DBDT-ABS*. In both *Priority Forwarding* and *DBDT-ABS*, we use the same distance-based formula, presented in Section 2, to calculate a host's waiting time and set the *Max_Wait_Time* to be the transmission radius. The latency of a flooding operation is then calculated as the time difference between the time when a host initiates a flooding packet to the time when the last host receives the packet. All these techniques guarantee flooding reachability. While *Simple Flooding* and *DBDT-ABS* do not require neighborhood knowledge, *Priority Forwarding* and *Self Pruning* require each host to know its 1-hop neighbors.

In each simulation run, we randomly choose a host and let it broadcast one data packet using different broadcasting techniques. Only one broadcast occurred at any one time. For each broadcast, we recorded the desired performance metrics and computed the corresponding average values. Each data point in our plots has a minimum confidence level of 95% and is the average of the results from 30 simulation runs generated as discussed above. Roughly, we simulated a campus-size network, a domain region about 10 to 100 hops.

A. Effect of Host Density

In this study, we fixed the transmission radius at 10 meters and generated a certain number of hosts, from 2500

to 25000, in each simulation run and placed them randomly on a square region of $500 \times 500 \text{ meter}^2$. In other words, we increased network density from 0.01 to 0.1 host/meter^2 . Figure 2 (a) shows the retransmission rate under four techniques. The worst performer is *Simple Flooding*, incurring 100% retransmission rate. The retransmission rate under *Self Pruning* increases as the network becomes denser. In this scheme, when a host receives a packet, it has to forward the packet unless all its 1-hop neighbors are included in the packet's header. Thus, it performs better in a sparse network. However, our simulation shows that only the hosts close to network borders can likely avoid retransmissions. As for *DBDT-ABS*, it performs the same as *Simple Flooding* when the host density is low. However, as the network becomes denser, it performs better and eventually outperforms *Self Pruning*. This shows that using static waiting time cannot reduce retransmission effectively, even though it does make flooding propagate in some order. In contrast to all other three techniques, *Priority Forwarding* incurs significantly lower retransmission rate. Figure 2 (a) shows that the retransmission rate becomes lower when the network becomes denser. Since the number of hosts participating in a flooding operation does not increase as much as the host density increases, the flooding cost under *Priority Forwarding* is not very sensitive to the network density. This study shows the effectiveness of dynamic waiting in eliminating redundant retransmission. Dynamic waiting, on the other hand, does not contribute latency to a flooding operation. As showed in Figure 2 (b), *Priority Forwarding* incurs much less flooding latency than *DBDT-ABS*. This is made possible with priority checking. Because it makes the hosts closest to the perimeter of a broadcast forward without delay, a flooding packet can be propagated as quickly as possible. We note that the network latency under *DBDT-ABS* decreases as the network becomes denser. This can be explained as follows. When a network becomes denser, each host has more neighbors within its transmission coverage and the host closest to the transmission perimeter should become closer. Thus, this host, which is the furthest, has to wait less and this make a flooding propagate faster.

B. Effect of Network Area

In this study, we increased the network area, from 100×100 to $1000 \times 1000 \text{ meter}^2$, in each simulation run and fixed the host density at $0.05 \text{ host/meter}^2$. The generated hosts are placed randomly on the network square domain. We fixed the transmission radius at 10 meters. Figure 3 (a) shows all four techniques have quite stable retransmission rates with respect to network size. The worst performer is *Simple Flooding* with 100% retransmission rate while the best one is *Priority Forwarding* with less than 30% retransmission rate. As for *Self Pruning* and *DBDT-ABS*,

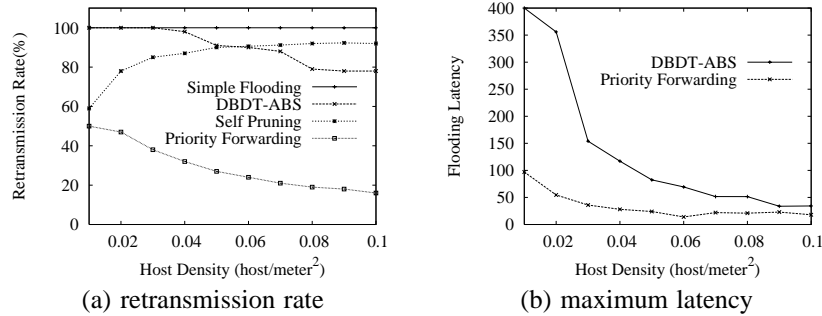


Fig. 2. Effect of Network Density

both require more than 85% of hosts to participate in packet forwarding. Note that Priority Forwarding and Self Pruning both take advantage of 1-hop neighborhood knowledge, but they have significantly different performance. This confirms our observation that a good order of flooding propagation, preserved by dynamic waiting, can help eliminate a large amount of redundant retransmission. Priority Forwarding also incurs much lower retransmission rate than DBDT-ABS. This performance gain comes from two factors. First, in Priority Forwarding, a host uses 1-hop neighborhood knowledge to determine if it should forward a packet. In contrast, in DBDT-ABS, a host does not have such knowledge and therefore, to avoid a retransmission, it has to make sure its transmission area is fully covered by the duplicate packets it receives. Our simulation shows that a host may receive a set of duplicate packets which together can cover all its 1-hop neighbors, but in most cases these packets cannot fully cover the host's entire transmission area. Second, Priority Forwarding dynamically adjusts a host's waiting time to collect more duplicate packets. It first seems that such dynamic extension of waiting could prolong flooding latency. However, this is avoided with our priority checking feature: the host furthest from a packet's sender has the highest forwarding priority and will forward the packet immediately. This feature significantly reduces the flooding latency. As Figure 3 (b) shows, the flooding latency under Priority Checking is much lower than that under DBDT-ABS.

C. Effect of Transmission Radius

In this study, we varied the transmission radius, from 5 to 15 meters, in each simulation run. The network area is fixed at $500 \times 500 \text{ meter}^2$ and the host density $0.05 \text{ host/meter}^2$. Figure 4 (a) and (b) show the retransmission rate and flooding latency, respectively, under the four techniques. Interestingly, the retransmission rate under Self Pruning increases as the radius increases. This can be explained as follows. Assume two hosts are within 1-hop to each other. When their radius increases, their overlapping

coverage increases. However, their non-overlapping portion also increases. Thus, it becomes less likely they have exactly the same 1-hop neighbors. This means that when a host receives a packet, it is more likely this host has to forward the packet, because Self Pruning allows one to drop a packet unless all its neighbors are also within 1-hop distance to the sender of this packet. The retransmission rate under DBDT-ABS drops as hosts increase their transmission radius, but it is far worse than Priority Forwarding in all cases. As for flooding latency, Figure 4 (b) shows that DBDT-ABS improves when transmission radius increases, although it is still much worse than Priority Forwarding. Such performance gain, however, mainly comes from the fact that increasing transmission radius reduces the number of hops of propagating a flooding packet to cover the entire network. This is because when the host density is fixed, increasing transmission radius does not affect the forwarding delay of those hosts who are nearest to the perimeter of a broadcast coverage.

V. CONCLUSION

We have presented a new flooding technique called *Priority Forwarding*, which has several advantages. First, it is highly efficient in reducing unnecessary packet retransmissions. By allowing a host to dynamically adjust its waiting time, the host can have better chance to collect more duplicate packets to avoid packet retransmission. Second, it minimizes flooding latency. With priority checking, the host closest to the coverage perimeter of a flooding packet will forward the packet immediately without delay. Thus, a flooding packet can be propagated as quickly as possible. Third, the new technique is simple and low-cost in implementation, as it requires each host to know only its distance to its 1-hop neighbors. Finally, the proposed technique allows a host to drop off a packet only when its 1-hop neighbors can receive the same packet from other hosts. Thus, it guarantees the flooding reachability.

REFERENCES

- [1] D. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. F. Korth, editors,

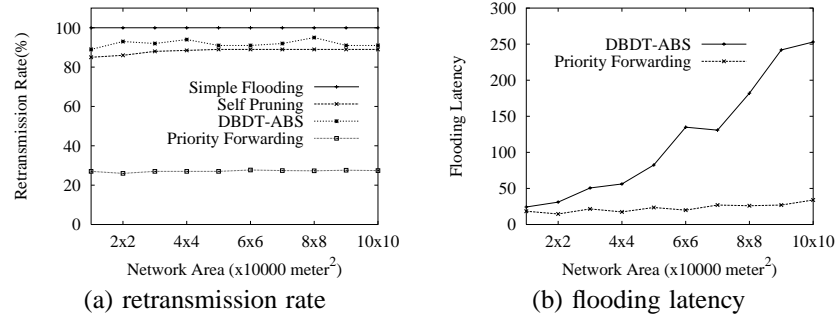


Fig. 3. Effect of Network Area

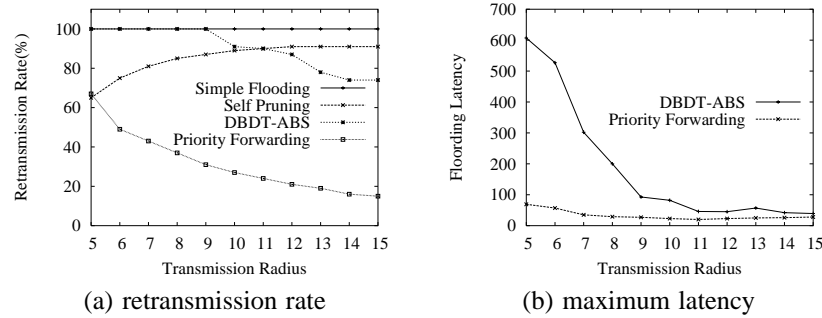


Fig. 4. Effect of Transmission Radius

- Mobile Computing*, pages 153–181. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.
- [2] C. E. Perkins. Ad Hoc On-Demand Distance Vector (AODV) Routing. INTERNET DRAFT - Mobile Ad hoc NETWORKING (MONET) Working group of the Internet Engineering Task Force (IETF), November 1997.
 - [3] Z. J. Haas and M. R. Pearlman. The Zone Routing Protocol (ZRP) for Ad Hoc Networks. INTERNET DRAFT - Mobile Ad hoc NETWORKING (MONET) Working group of the Internet Engineering Task Force (IETF), November 1997.
 - [4] Y. Ko and N. Yaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proc. of MOBICOM'98*, pages 66–75, 1998.
 - [5] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for Reliable Multicast in Multi-hop Ad Hoc Networks. In *Proc. of the Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication*, pages 64–71, 1999.
 - [6] J. Jetcheva, Y. Hu, D. Maltz, and D. Johnson. A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks. Internet Draft: draft-ietf-manet-simple-mbcast-01.txt, July 2001.
 - [7] E. D. Kaplan. *Understanding GPS: principles and applications*. Artech House, Boston, MA, 1996.
 - [8] S. Ni, Y. Tseng, Y. Chen, and J. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proc. of MOBICOM'99*, pages 151–162, 1999.
 - [9] Y. Tseng, S. Ni, and E. Y. Shih. Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Networks. In *Proc. of ICDCS'01*, pages 481–488, 2001.
 - [10] Y. Sasson, D. Cavin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Swiss Federal Institute of Technology, Technical Report IC/2002/54*, 2002.
 - [11] H. Lim and C. Kim. Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks. In *Proc. of the ACM Int'l Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM)*, pages 61–68, August 2000.
 - [12] W. Lou and J. Wu. On Reducing Broadcast Redundancy in Ad Hoc Wireless Networks. *IEEE Transactions on Mobile Computing*, 1(2):111–123, 2002.
 - [13] W. Peng and X. Lu. AHBP: An Efficient Broadcast Protocol for Mobile Ad Hoc Networks. *Journal of Science and Technology, Beijing, China*, 2002.
 - [14] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS'2001)*. IEEE Computer Society, 2001.
 - [15] I. Stojmenovic, M. Seddigh, and J. Zunic. Dominating Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 13(1):14–25, January 2002.
 - [16] J. Sucec and I. Marsic. An Efficient Distributed Network-Wide Broadcast Algorithm for Mobile Ad Hoc Networks. In *Rutgers University, CAIP Technical Report 248*, September 2000.
 - [17] J. Wu and F. Dai. Broadcasting in Ad Hoc Networks Based on Self-Pruning. In *Proc. of INFOCOM'03*, March 2003.
 - [18] M. T. Sun, W. C. Feng, and T. H. Lai. Location Aided broadcast in wireless ad hoc networks. In *Proc. of GLOBECOM'01*, 2001.
 - [19] Chun-Chuan Yang and Chao-Yu Chen. A Reachability-Guaranteed Approach for Reducing the Broadcast Storms in MANETs. In *IEEE Semiannual Vehicular Technology Conference (VTC-2002)*, to appear, 2002.
 - [20] R. Want and A. Hopper. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.
 - [21] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications Magazine*, 4(5):42–47, 1997.
 - [22] P. Castro, P. Chiu, T. Kremenek, and R. Muntz. A Probabilistic Room Location Service for Wireless Networked Environments. In *UbiComp2001*, pages 18–34, 2001.
 - [23] C. Randell and H. Muller. Low cost indoor positioning system. In *UbiComp2001*, pages 42–48, 2001.
 - [24] W. Peng and X. Lu. On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks. In *Proc. of MOBIHOC'00*, 2000.