Exploring the Energy-Latency Trade-off of Broadcasts in IEEE 802.11 Power Save Networks

Matthew J. Miller
mjmille2@uiuc.edu

Cigdem Sengul
sengul@uiuc.edu

Abstract

Efficient information dissemination is essential for the operationality of sensor networks. However, one of the challenges to efficient information dissemination arises from only a random subset of nodes being active in any given time. This is mainly due to sensor networks employing power-saving techniques to compensate the energy constraints of sensor nodes. Specifically, significant delay is incurred from any packet transmission attempt in a power-saving network that allow nodes periodically switch to sleep modes to conserve energy. In this paper, we propose a Probability-Based Broadcast Protocol (PBBF) that aims to provide fast broadcast propagation in the case of such networks. Simulation studies show that PBBF efficient and effective information dissemination with low latency while still maintaining high delivery ratios.

1. Introduction

Sensor networks are likely to be widely deployed due to their potential use in sensing inhospitable and inaccessible physical environments. In such networks, efficient information dissemination among sensors may be necessary for communicating data to all sensors in the network. In particular, this can provide a consistent view of data at each sensor and increase the fault-tolerance of the system. As an example, once deployed the sensor network should be reprogrammable [9]. Therefore, it may be critical to efficiently disseminate a software update to all sensor nodes.

Information dissemination effectively depends on an efficient broadcast mechanism. To this end, we propose a Probability-Based Broadcast Forwarding (PBBF) protocol for sensor networks. We assume a sensor network where nodes periodically switch to a sleep mode to conserve energy and broadcast messages are probabilistically sent without waiting for all the nodes in the neighborhood of the sender to wake up. In particular, PBBF takes advantage of the high node density in sensor networks and assumes nodes that are in sleep modes during a broadcast eventually receive the message via some other neighbor. The key benefit of our approach is fast broadcast propagation in the presence of sleeping nodes.

In addition to sensor networks, PBBF can also be applied to general ad hoc networks, where mobile wireless nodes form a network without any fixed infrastructure. A major focus of ad hoc networking research is the design of *proactive* and *reactive* routing protocols. In reactive protocols a route is computed to a destination only when it is needed. However, this is typically done via a network-wide flooding. Therefore, on-demand routing protocols can also benefit from our scheme. Specifically, PBBF provides efficient flooding, which potentially results in fast route discovery in power saving networks.

The rest of the paper is organized as follows. Section 2 presents related work. In Section 3, we describe our proposed protocol. Simulation results are presented in Section 4. Section 5 concludes our paper and addresses future work.

2. Related Work

This section presents power saving protocols for ad hoc networks. We further discuss two broadcast applications in ad hoc networks: *Ad hoc routing* and *Code dissemination*.

2.1. Power Save in Ad Hoc Networks

Our work primarily focuses on modifying the Power Save Mechanism (PSM) in IEEE 802.11 [7]. In IEEE 802.11 PSM, nodes are assumed to be synchronized and wake up at the beginning of each beacon interval. They remain on for the duration of an ATIM (Ad-hoc Traffic Indication Map) window, during which a node may advertise packets in its link layer queue via an ATIM packet. Assuming the ATIM is for a unicast packet, the destination will respond with an ATIM-ACK packet. If the ATIM is for a broadcast packet, no ATIM-ACKs are sent. At the end of the ATIM window, a node will remain on if it received an ATIM, an ATIM-ACK, or advertised a broadcast ATIM. Otherwise, a node will return to a sleep state until the next beacon interval to conserve energy. Packets which are unable to be advertised during the ATIM window or arrive after the ATIM window cannot be sent until after the next beacon interval. The success of IEEE 802.11 PSM depends on ensuring that nodes that want to communicate with each other are awake at the same time to coordinate notification of packets and the receiver remains awake to receive the pending transmission.

Figure 1 illustrates PSM for three nodes within range of each other. The dotted arrows denotes a “causes” relationship. **A** has a packet for **B** at time

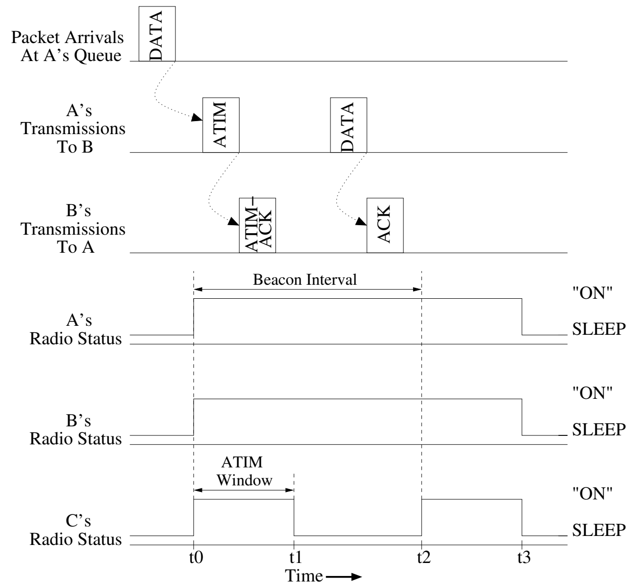


Figure 1. 802.11 Power Save Mode

t_0 and therefore sends at ATIM during the ATIM window. **B** replies with an ATIM-ACK. After the ATIM window, but before the next beacon interval, **A** transmits the data packet to **B** and receives an ACK. Note that **A** and **B** must remain on until the next beacon interval, even if all data packets have been sent.

An immediate observation about PSM is that latency is greatly increased for multihop flows. At each hop, the packet must wait at least one beacon interval, which is much longer than the time it would take to immediately transmit the packet. This also affects throughput and packet loss when the network is dynamic due to factors such as mobility. Because the latency is increased, there is a greater probability that the chosen route will break before the packet reaches its destination. This problem has been addressed in previous work by selectively choosing a small subset of nodes which always remain on between beacon intervals [14, 1]. The main goal in this approach is to create a connected dominating set (CDS), which serves as a “routing backbone” in the network (i.e., all packets are routed through the backbone). All nodes on the CDS remain active all the time to maintain global connectivity, whereas all other nodes are in power-save mode (e.g., IEEE 802.11 PSM).

In GAF [14], nodes are assumed to have GPS capabilities and form a virtual grid. The key aspect of this grid is that the size of the grid boxes is chosen such that any pair of nodes in adjacent grid boxes are within transmission range of each other. Thus, only one node per grid box needs to remain on for forwarding while the others can conserve energy. A disadvantage of GAF is packets may have to traverse more hops since some nodes in a given grid box may be able to transmit to nodes beyond the next adjacent grid box.

SPAN [1] achieves similar results by allowing nodes to advertise information such that an approximation of a connected dominating set is formed where each node is within one hop of a *coordinator* node. The coordinator nodes do not enter PSM and hence a packet can be routed through paths of coordinators to avoid excessive latency. Both GAF and SPAN use heuristics to allow nodes to rotate the roles of sleeping and remaining on such that energy usage remains fairly balanced.

In On-Demand Power Management [15], an on-demand routing protocol is assumed. When a node sets up a route or forwards data, it will remain on and set a timer to return back to PSM. This timer is refreshed whenever data is forwarded through a node. If the timer expires, the node returns to PSM. Thus, nodes on active routing paths will remain on to achieve low latency, while those not actively forwarding data will conserve energy via PSM.

However, each of these works only addresses unicast packets, whereas our work is concerned with reducing the latency of broadcast packets. All of these protocols, as described, would have to revert back to 802.11's high-latency PSM protocol to send broadcast packets.

2.2. Broadcast Applications in Ad Hoc Networks

We briefly describe two types of applications for ad hoc networks which utilize broadcasting and hence could benefit from our proposed scheme. These applications are on-demand ad hoc routing and code dissemination in sensor networks.

Vast amounts of research have gone into ad hoc routing. However, the two most widely used protocols [8, 11] are both fundamentally similar and differ only in details that are not relevant in our current context. Both use a similar route discovery process consisting of Route-Request (RREQ) and Route-Reply (RREP) packets. The details of this process are described in detail in the aforementioned papers.

The basic behavior of ad hoc routing is a source node starts a *Route Discovery* by doing a broadcast flood of *Route-Request* (RREQ) packets to find a specified destination for whom the source has data to send. When the RREQ reaches the specified destination or an intermediate node with cached information about the destination, a *Route-Reply* (RREP) packet is unicast back to the source. Upon receiving the RREP, the source can begin sending data on the path. This procedure may need to repeat as links break and new paths need to be discovered.

The broadcast nature of the route discovery usually results in nodes receiving multiple, redundant copies of a RREQ. To reduce the overhead from such flooding of RREQ packets, in gossip-based ad hoc routing [5] each node forwards a routing message with some probability. Our approach is similar to gossip-based ad hoc routing in the sense that the nodes decide to broadcast a message either immediately or at a later time with some probability, which has an effect on the number of nodes receiving the broadcast. However, our concern is energy-saving networks, where nodes are in sleep mode periodically, whereas gossip-based ad hoc routing has been evaluated in networks where all nodes are in active mode. We believe that PSM will greatly affect the latency of route discovery.

Another broadcast application involves distributing code updates among sensor nodes. In particular, if envisioned sensor networks can operate for weeks or years, presumably updates will need to be periodically applied to the software running on the sensor. This can be done by broadcasting patches throughout a sensor network. In [12], different methods are explored for applying such updates at a system level in sensor networks.

Efficient information dissemination problem is also addressed by SPIN [6] in a different context. All individual sensor observations are propagated to all sensor nodes across the network. SPIN protocols incorporate *negotiation* and *resource adaptation* in order to avoid deficiencies of the classic flooding approach. These deficiencies include *implosion* (i.e., data is sent regardless of whether or not the neighbor has already received the data), *overlap* due to sensors covering overlapping regions, and *resource blindness* (i.e., the activities are not modified based on the amount of energy). We believe our approach partly overcomes the implosion problem by neighbors selectively broadcasting information without the expensive negotiation via *advertisement* and *request* messages in SPIN. While overlapping information is not our concern, energy-awareness can be applied to our design by allowing nodes with limited remaining energy to resign from update propagation responsibility.

3. Probability-Based Broadcast Forwarding

We propose using Probability-Based Broadcast Forwarding (PBBF) to improve the latency of broadcasts in PSM networks. Our approach presents a trade-off between energy, latency, and the fraction of nodes that receive the broadcast information. Because of the flooding nature of broadcasts, nodes potentially receive multiple, redundant copies of a broadcast packet. PBBF exploits this redundancy in broadcast communication and forwards packets using a probability-based approach. Our aim is to ensure, with high probability, a node receives at least one copy of each broadcast packet.

Our protocol is based on a simple idea of adding a parameter, p, to 802.11 PSM. This parameter is the probability a node rebroadcasts a packet in the current beacon interval despite the fact that the packet has not been advertised with an ATIM. With probability $(1-p)$, the node waits to rebroadcast the packet after it is advertised in the next ATIM window. If a node chooses to rebroadcast the packet immediately, only the subset of neighbors which are currently awake can receive the packet, but with no PSM-induced delay. However, if the node chooses to wait to do the rebroadcast, it is virtually guaranteed that its entire neighborhood receives the packet. This guarantee comes at the expense of a potentially large delay that is induced due to waiting for next ATIM interval. Thus, 802.11's original PSM is a special case of PBBF with $p = 0$. The p parameter presents a trade-off between latency and the number of nodes receiving a broadcast. As p increases, latency decreases while the fraction of nodes not receiving a broadcast potentially increases. However, our experiments show that PBBF is successful in delivering a high percentage of broadcasts to all nodes. This is due to the fact that nodes typically receive duplicate broadcast packets from different neighbors. Therefore, with the PBBF approach, nodes eventually receive a broadcast from one of their neighbors.

The effectiveness of PBBF relies on nodes further away from the broadcast source (in terms of hop count) occasionally being awake when a closer node decides to broadcast a packet immediately. Since in IEEE 802.11 PSM nodes stay awake only to receive pending traffic, PBBF is significantly affected by the traffic distribution in a network. For example, if the load is low and few broadcasts are propagating through the network, regardless of the p value, when a node chooses to rebroadcast a packet immediately, with high probability, no neighbors will receive this packet for the first time. However, if the traffic is dense, this can potentially lead to increased congestion and queuing delays, thereby inducing second-order effects which can degrade overall performance. For simplicity, and to avoid such second-order effects, we add a second parameter, q, to PBBF. This parameter represents the probability a node remain on after the ATIM window even if it have not sent or received any ATIMs. Thus, PSM is still a special case of PBBF with $p = 0$ and $q = 0$. The "always on" mode (i.e., PSM is turned off) could be approximated by setting $p = 1$ and $q = 1$. PBBF would still be slightly different than "always-on" in this case because it would still have the byte and time overhead of the ATIM window.

From its description, it is clear that q represents a trade-off in terms of energy and the number of nodes receiving a broadcast. As q increases, energy consumption increases, but the fraction of nodes not receiving a broadcast decreases (for a fixed p value). Thus, by specifying these two parameters, we can investigate the trade-off between energy, latency, and the percentage of nodes receiving a broadcast.

3.1 Optimizations

There are a few heuristics we believe could further enhance PBBF. However, due to the time constraints of the project, these heuristics are not tested in our simulations. We leave detailed analysis of these heuristics as future work.

Overheard ATIM Traffic As the number of ATIM and ATIM-ACKs overheard in the current ATIM window increases, p can be increased. Intuitively, the number of ATIM and ATIM-ACKs overheard signify that more neighbors will be on after the ATIM interval and hence a larger fraction of nodes would receive the packet if it is rebroadcast immediately. Thus, the latency of the current interval can be reduced with less degradation in the number of nodes receiving the broadcast.

Reduced Overlap This heuristic assumes some type of measurement is available for a receiver to determine its physical distance from the sender (e.g., SINR measurements). When the receiver is close to the sender of the broadcast, the receiver decreases p. When two nodes are close, there is more overlap in the number of nodes who have received the packet from the sender and thereby, receive a duplicate if the receiver chooses to rebroadcast it. Therefore, if a packet is advertised, not much is gained by resending the packet immediately since most of its neighbors have already received the packet. This problem is similar to the overlap deficiency discussed in SPIN [6].

Remaining Time in the Beacon Interval As the amount of time until the next beacon interval decreases, p can be decreased. In this situation, a node achieves a better delivery fraction with a small increase in latency. However, if the broadcast packet is received long before the next beacon interval, the better delivery fraction has a much larger latency cost. The function by which p decreases could be chosen based on the relative importance of latency and delivery fraction in the network.

4. Simulation Results

The goal of our simulation study is to measure our success in meeting the design goals of PBBF and investigate the trade-off between energy, latency, and the percentage of nodes receiving a broadcast.

We implemented PBBF and 802.11 PSM using *ns-2* [10] network simulator. The DSR (Dynamic Source Routing) protocol that we use in our ad hoc routing simulations already exists in *ns-2*. In addition to these MAC and physical layer modifications, we implemented an application to simu-

Table 1. Radio Power Consumption (Watts)

P_{TX}	P_{RX}	P_{IDLE}	P_{SLEEP}
1.40	1.00	0.830	0.130

Table 2. Code Distribution Parameter Values

Parameter	Value
N	50
k	1 update(s)/broadcast
q	0.25
Δ	10.0
λ	0.2 packet(s)/s
μ	128 bytes
σ	64 bytes
x_{min}	32 bytes
x_{max}	256 bytes
BI	100 ms
AW	20 ms

late code distribution in a static (e.g., sensor) network. In Section 4.1, we present the simulations for broadcast in a code distribution environment. Section 4.2 analyzes the simulation results for the ad hoc routing experiments.

4.1. Code Distribution Application

We implemented 802.11 PSM at the MAC layer of *ns-2*. However, our implementation does not handle synchronization of nodes. Because PSM's time synchronization mechanism is only designed for single-hop networks and synchronization in multihop networks is a hard problem for which no good solutions exist [3], we assume perfect synchronization in the network. The length of the beacon interval, BI, and ATIM window, AW, are set according to the values in Table 2.

All nodes communicate with half-duplex wireless radios that conform to IEEE 802.11-based WaveLAN wireless radios with a bandwidth of 2Mb/s and a nominal transmission range of 250m. We use the same energy model as in [1], which is shown in TTable 1.

We implemented the broadcast application at the routing layer of *ns-2*. The protocol is relatively simple. One random node is chosen to be the broadcast and code distribution source for each scenario. Each broadcast packet contains the k most recent updates generated at the source. Thus, nodes do not need to receive every broadcast as long as they receive about $\frac{1}{k}$-th of the packets. The k parameter represents a trade-off in byte overhead versus the number of packets missed by a node.

Each non-source node keeps track of the sequence number of updates it has received. Optionally, our application allows negative acknowledgments (NACKs) to be unicast to the source when there is a gap in the sequence numbers of the updates a node has received. Each packet from the source contains updates with sequence numbers from seq_{low} to seq_{high}, where $seq_{high} - seq_{low} < k$. Thus, if a non-source node receives a broadcast packet and is missing some updates with a sequence number less than seq_{low}, it can unicast a NACK back to the broadcast source and the source responds with a NACK-REP packet which includes the missing updates. Since *ns-2* does not support 802.11 fragmentation, we limit the number of updates in a NACK-REP packet to be the k oldest updates requested.

For a broadcast message, no routing protocol is necessary. However, the unicast NACKs and corresponding NACK-REP packets require some routing functionality to test multihop scenarios. Because the effects of our PSM proposal on ad hoc routing are tested in a separate experiment, we did not want residual effects of the routing protocol to appear in the broadcast application experiments. Therefore, the routing tables for nodes are "hard-wired" when a scenario is created. Since the scenarios are static, we run Floyd-Warshall's all-pairs shortest path algorithm after the nodes have been placed to manually configure the routing tables. Thus, there is no routing overhead in these experiments.

For update generation, new updates are generated and sent deterministically at the source at a rate of λ updates/second. The byte size of a new update is chosen according to a truncated normal distribution with mean μ, standard deviation σ, a minimum value x_{min}, and a maximum value x_{max}. The simulation values for these parameters are shown in Table 2.

To test PBBF in this setting, in addition to varying the p and q values, we also changed the k value and the network density, Δ. To define Δ, we use the following equation:

$$\Delta = \frac{\pi R^2 N}{A} \tag{1}$$

where R is the range of a node (250 m), N is the number of nodes, and A is the area of the region where nodes are located. For our simulations, we fixed N and changed A to get the desired Δ. The fixed value of N is shown in Table 2. Also, in Table 2, the k, q, and Δ parameters show the fixed value when that particular parameter is not changed. For example, when q is being varied on the x-axis, k and Δ are fixed at the values in Table 2.

We ran each simulation for 500 seconds and each data point is averaged over ten runs. The error bars show the standard deviation of the data. For brevity, we omit the results which include NACKs in the protocol. Therefore, if a node does not receive a particular update broadcast, it is missed.

The impact of q parameter Our first experiments show show various values of q affect PBBF. Recall that q is essentially a toy parameter in our protocol which could emulate how often nodes are waking up for other traffic in the network. Figure 2 shows how the average energy consumed at a node, normalized for the number of updates generated, changes with q. We can see that using PSM saves almost 3 Joules per update over using no PSM. The figure also shows that energy

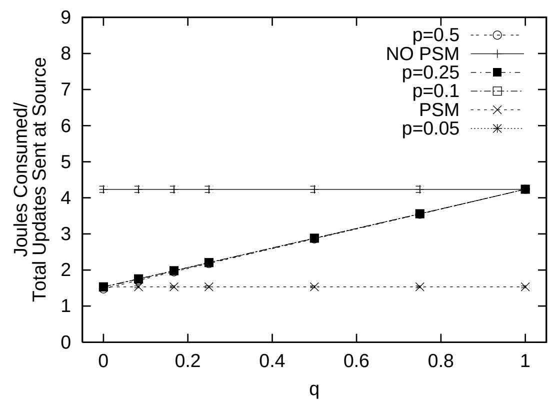
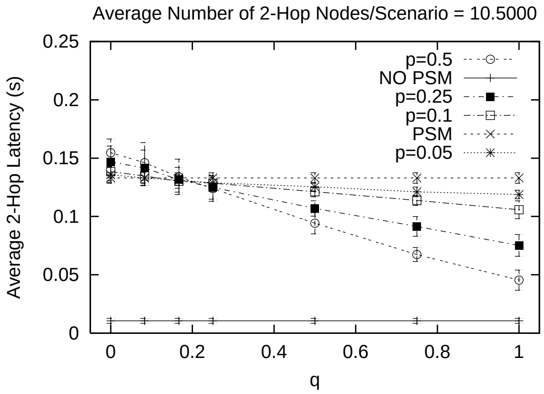


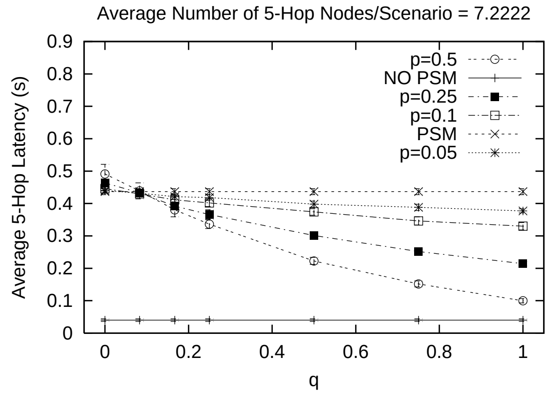
Figure 2. Average Energy Consumption

increases linearly with the q value. We also observe that q dominates p in the energy usage because regardless of the p value, the PBBF lines overlap.

In Figure 3, we see the impact of q on the latency. Figure 3(a) and Figure 3(b) show the average latency of nodes that are two hops and five hops from the source, respectively. In our simulations, new packets always arrive at the code source during the ATIM window, so they are sent with a delay of about AW. As expected, the latency to reach two hop neighbors is about $AW + BI$. We can see that PSM consistently has a high latency, whereas turning PSM off results in a much lower latency. PBBF does worse than PSM at small values of q, but improves significantly as q and p increase. The reason PBBF performs worse for small values of q is the amount of redundancy in broadcasts received from different neighbors is reduced. Therefore, it is more likely that a node will not receive the broadcast from the neighbor which would result in the smallest latency. However, as q and p get larger, there is a greater chance a broadcast will be transmitted and received without waiting for the next beacon interval. From the Figure 3, we can also see that the cross-over q point where PBBF does better



(a) 2-Hop Latency



(b) 5-Hop Latency

Figure 3. Average Update Latency

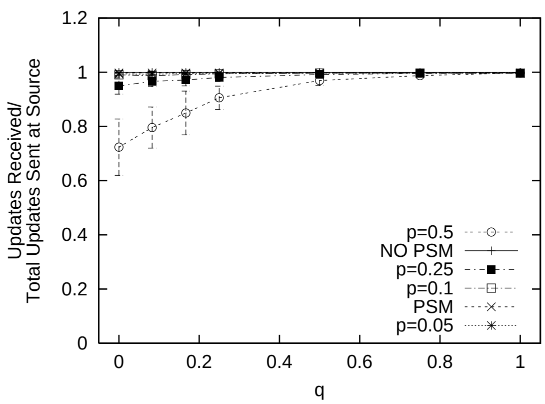


Figure 4. Average Updates Received

than PSM occurs at a lower value for nodes farther from the source. This is expected since there is a greater probability that at least one node between the source and a distant node will be able to reduce the latency by a beacon interval. Also, there are potentially many more different paths by which the broadcast can reach distant nodes.

Figure 4 illustrates how the q value affects the fraction of updates a node receives. We observe that setting $p = 0.5$ results in a significant degradation until q reaches about 0.5. For $p = 0.25$, there is a little degradation and all the other p values result in less than 1% loss.

The impact of k parameter We next investigate the effects of placing more updates in each broadcast packet as described earlier in this section. We use the fixed q value in Table 2 in these experiments. For brevity, we omit the energy consumption because it remained constant regardless of the k value. The Joules consumed per update for PSM, PBBF, and no PSM were about 1.5, 2.25, and 4.25, respectively.

Figure 5 shows that k can significantly affect the latency of PBBF. Increasing k linearly adds more per packet byte overhead. However, the major effect on update latency in PBBF comes when an update is missed the first time, but received in a later broadcast. Thus, we can see a large jump from $k = 1$ to $k = 2$ for $p = 0.5$. This is because, when $p = 0.5$, a significant fraction of the updates are not received when they are initially broadcast. For example, an update U_n may be missed when it is initially broadcast, but received whenever update U_{n+1} is broadcast. Thus, the fraction of updates received is improved because U_n would not have been received if $k = 1$. However, U_n incurs an increase in latency of about $\frac{1}{\lambda}$ since it iz received in the next broadcast generated at the source.

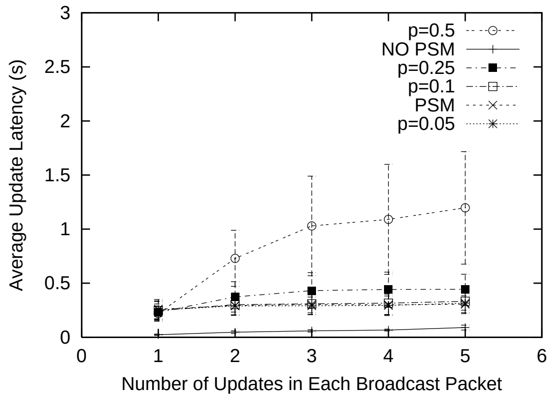


Figure 5. Average Update Latency

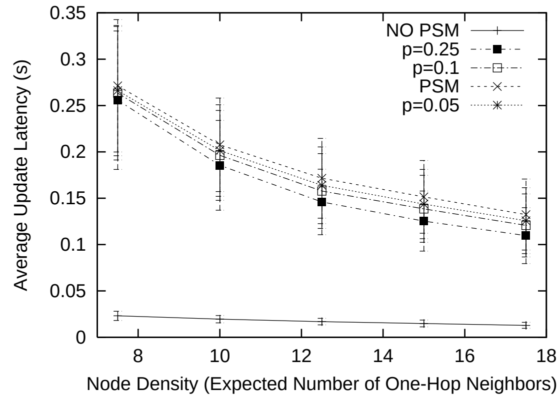


Figure 7. Average Update Latency

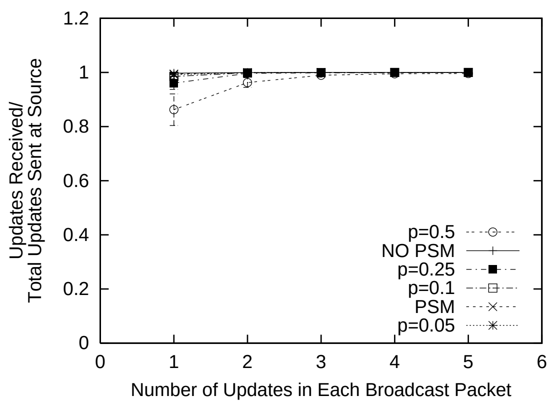


Figure 6. Average Updates Received

Figure 6 shows how increasing k can improve the fraction of updates received. When $k = 3$, virtually every update is received in each protocol. At $k = 1$, we observe that for p values of 0.25 and 0.5 PBBF results in noticeable degradation in the number of updates received. From Figure 5 and Figure 6, we can conclude that increasing k will improve the delivery fraction, but at the expense of increased average latency for updates.

The impact of Δ Finally, we investigate how the density of the network affects the protocols. The node density, Δ, is about equal to the expected number of one-hop neighbors for a node. From Figure 7, we see that latency shows improvement as Δ increases since nodes are expected to be fewer hops from the source. This has the most drastic effect on PSM and PBBF since nodes wait less beacon intervals before receiving an update. The effect on PSM and PBBF appears to be about the same with neither showing much improvement relative to the other as Δ changes. Figure 8 illustrates that PBBF does better with respect to the number of updates received as Δ increases. This is intuitive since increasing Δ increases the probability of redundant broadcasts a node can receive. We omit the the energy consumption graphs for brevity but note it stays relatively constant regardless of the Δ value. The energy values are the same as those for the simulations that evaluate the effect of k, which is discussed previously.

4.2. Ad-Hoc Routing

In this section, we investigate the effects of PBBF on an ad hoc routing protocol, DSR [8]. The effects of PBBF on ad hoc routing can be evaluated by its impact on the following properties:

- *Packet delivery ratio:* The ratio of data packets delivered to the destinations to those generated by the sources.

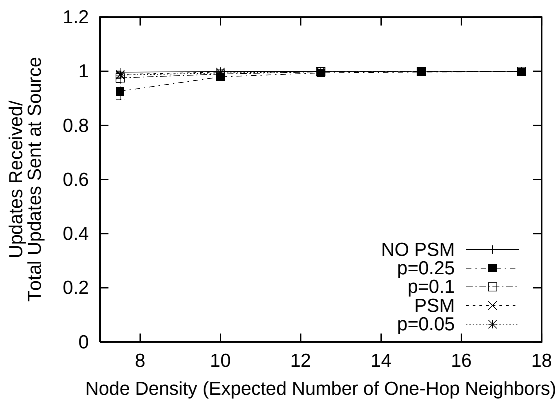


Figure 8. Average Updates Received

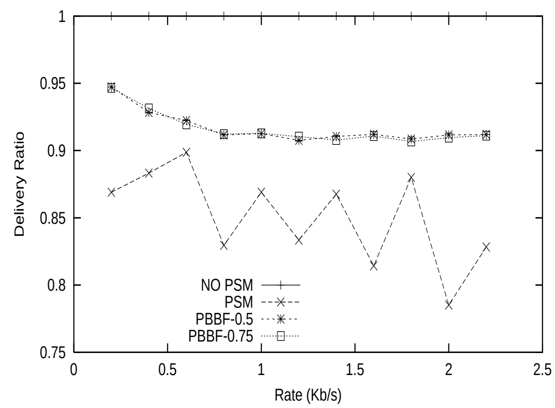


Figure 9. Delivery Ratio

- *Average delay for source to discover a route to a destination:* The difference between the time a RREQ packet sent by the sender and the time a RREP was received by the sender, including al the delays due to route discovery, queueing delays, retransmission delays at the MAC layer, propagation times and delays incurred by power saving mechanism (e.g., the delays from buffering packets for sleeping nodes).

- *Energy goodput (EG):* Energy used per unit data delivery [15], which is defined as follows:

$$EG(bit/J) = \frac{total\_bits\_transmitted}{total\_energy\_used} \quad (2)$$

- *Average hop count:* Average number of hops a data packet traverses to reach its destination.

- *Sleep time:* Average time the nodes spend in sleep mode.

4.3 Results

To evaluate the performance of PBBF with varying traffic load, we simulate PBBF in 500mx500m network with 60 nodes with long-lived CBR connections at different transmission rates between 0.2Kb/s-2.2Kb/s. All data packets are of length 128 bytes. There are 20 connections started randomly between 20s and 25s. Each simulation runs for 600s. There is no mobility in the network. Our simulation results represent an average of five runs with identical traffic models, but different randomly generated network topologies.

We evaluate PBBF with p=0.5 and p=0.75, and q = 0, to see how simple PBBF algorithm performs compared to PSM and NO PSM. Our main goal is to show that even with in its simplest form, PBBF can achieve performance improvements compared to PSM.

Figure 9 shows NO PSM has 100% delivery ratio for all traffic loads. On the other hand, from the figure, we observe that althoug PSM delivers 80%-90% of packets, it has the most unpredictable behavior. However, with PBBF-0.5 and PBBF-0.75, the delivery ratio performance increases by 10% and PBBF exhibits more stability. We believe, this is due to PBBF's success in distributing the traffic load over a longer time period. Specifically, with PSM, all traffic should be advertised in a limited ATIM window, which may cause high loss of packets, and hence low delivery ratios.

Figure 10 shows that PBBF has the best energy goodput (i.e., more bits are delivered per joule

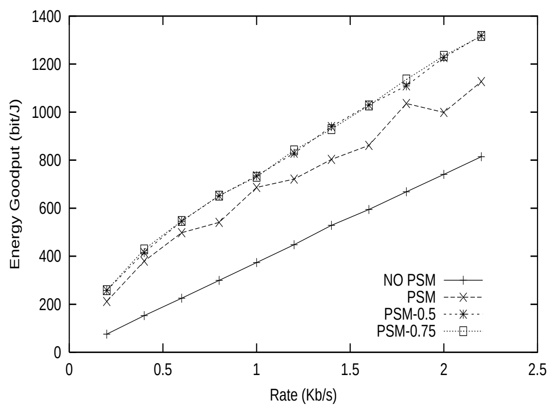


Figure 10. Energy Goodput

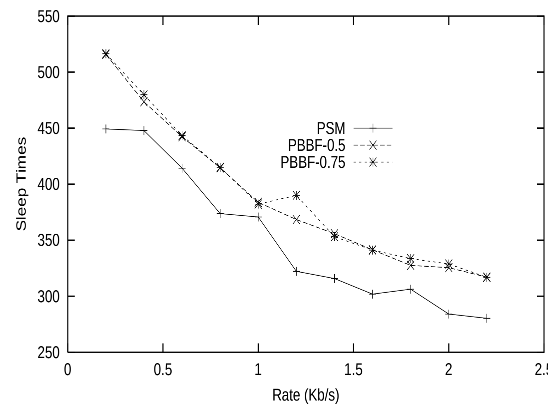


Figure 11. Average Sleep Times

Table 3. Average Hop Count

	NO PSM	PSM	PSM-0.5	PSM-0.75
1	1.4	1487	1.444	1.441
2	1.2	1.245	1.25	1.248
3	1.2	1.536	1.538	1.531
4	1.67	1.725	1.735	1.742
5	1.5	1.529	1.537	1.535

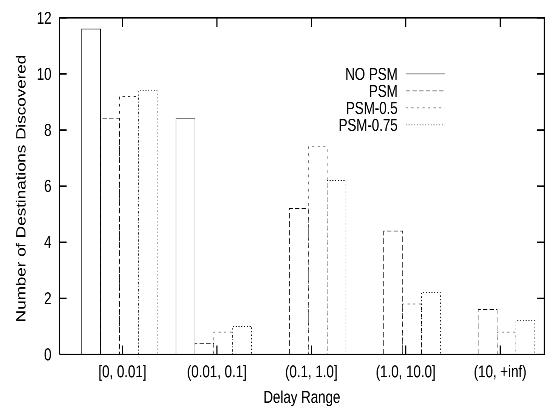


Figure 12. Delay Distribution for Destination Discovery

compared to PSM and NO PSM). As expected, NO PSM shows the worst performance since each node is in active mode at all times. The main reason why PBBF outperforms PSM is illustrated in Figure 11. From the figure, we see that PBBF allows nodes to stay in sleep modes for considerably longer times (e.g., an average of 50s more in 600s simulation time). Therefore, PBBF achieves better performance by not waking up all nodes for each broadcast forwarding in terms of both delivery ratio and energy goodput. Furthermore, PBBF achieves higher energy goodput although it uses slightly longer hops, which confirms that energy savings due to longer sleep times are significant (see Table 3 for 5 different scenarios).

Additionally, we also observe improvements in delay performance. From Figure 12, DSR with PBBF is able to find routes to destinations faster compared to PSM (though not significantly). However, in our simulations, there have been cases where DSR with PSM could not discover routes for some connections (i.e., some connections had 0% throughput due to not finding a route). There have been no such cases when DSR runs with NO PSM or PBBF.

In summary, delivery ratio, energy goodput, sleep times and delay results confirm our expectations that PBBF achieves performance improvements compared to basic PSM approach in an ad hoc routing environment.

5. Conclusion

Latency of broadcast propagation in power-saving networks may significantly increase due to waiting for all the nodes to wake up to receive the transmission. In this paper, we present probabilisty-based broadcast forwarding (PBBF) that reduces the latency of broadcast propagation in such networks. Simulations show that PBBF is able to improve delay performance without degrading the throughput (i.e., a high fraction of nodes receive the broadcast). Specifically, PBBF is successful in exploiting the trade-off among latency, energy consumption and throughput. We expect that various optimizations discussed in Section 3.1 applied to PBBF can also provide futher improvements. Our future plan is to investigate the benefits of these heuristics in various network topologies. Additionally, we will study the critical probabilities for PBBF using percolation model in those topologies to obtain a bound on critical probabilities.

References

[1] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *ACM MOBICOM 2001*, July 2001.

[2] O. Dousse, P. Thiran, and M. Hasler. Connectivity in ad-hoc and hybrid networks. In *IEEE INFOCOM*, pages 1079–1088, New York, 2002.

[3] J. Elson and K. Römer. Wireless Sensor Networks: A New Regime for Time Synchronization. In *ACM Hot Topics in Networks 2002*, October 2002.

[4] G. Grimmet and A. M. Stacey. Critical probabilities for site and bond percolation models. *Annals of Probability*, 26:1788–1812, 1998.

[5] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-Based Ad Hoc Routing. In *IEEE INFOCOM 2002*, June 2002.

[6] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive Protocols for Information in Wireless Sensor Networks. In *ACM MOBICOM 1999*, August 1999.

[7] IEEE 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[8] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[9] P. Levis and D. Culler. Mate: A Tiny Virtual Machine for Sensor Networks. In *ASPLOS 2002*, 2002.

[10] `ns-2` Network Simulator. `www.isi.edu/nsnam/ns`.

[11] C. E. Perkins and E. M. Royer. Ad-Hoc On Demand Distance Vector Routing. In *IEEE WMCSA 1999*, February 1999.

[12] N. Reijers and K. Langendoen. Efficient Code Distribution in Wireless Sensor Networks. In *ACM WSNA 2003*, September 2003.

[13] Y. Sasson, D. Carin, and A. Schiper. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *IEEE Wireless Communication and Networking Conference (WCNC)*, March 2003.

[14] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed Energy Conservation for Ad Hoc Routing. In *ACM MOBICOM 2001*, July 2001.

[15] R. Zheng and R. Kravets. On-demand Power Management for Ad Hoc Networks. In *IEEE INFOCOM 2003*, April 2003.

6 Appendix - Discussion on PBBF's Behavior Using Percolation Theory

PBBF algorithm is in some sense a gossiping protocol, which can be formulated as:

- The source advertises a broadcast in the ATIM window with probability 1.
- When a node receives the broadcast, with probability p it broadcasts withhout waiting for the next ATIM window and with probability $(1 - p)$ the node waits for the next ATIM window to advertise the broadcast.

However, gossiping in ad hoc networks have different characteristics than traditional gossiping. Specifically, in previous research in epidemics, it is assumed that any node in the network can send a message to any other node. Each node gossips by choosing a set of nodes at random to which to gossip. However, in ad hoc networks such an assumption cannot be made. In an ad hoc network, due to the characteristics of the radio communication, the message is received by all nodes in close proximity of the sender (i.e., one-hop neighbors of the sender) [5]. If the source has relatively few neighbors, there is a chance that the gossiping (in our case, the broadcast) will die. If we think this kind of gossiping is an epidemic, then the spreading of the epidemic depends on how many people each person can infect. Therefore, the likelihood of the propogating a broadcast depends on how the nodes are placed in the network (i.e., the characteristics of the topology graph). Additionally, in PBBF how many nodes can receive the broadcast is a factor of when the broadcast is sent. Specifically, if the node sends the broadcast without waiting for the ATIM, the number of nodes that can receive the broadcast, n_{recv}, satisfies $0 \leq n_{recv} \leq n_{awake}$, where n_{awake} is the number of one-hop neighbors that are awake during the broadcast. In the worst case, assuming no traffic, the average $n_{awake} = q \times d(sender)$, where $d(sender)$ is the degree of the sender (i.e., number of one-hop neighbors of the sender) and q, as mentioned earlier, is the probability that a node stays awake even though there is no traffic. Based on this information, the main question is "Under what conditions would a broadcast die out?"

We believe an answer can be found using percolation theory. Percolation theory states that 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. In PBBF, N is the set of nodes that send the broadcast without waiting for the ATIM window, and the message is not received by any of the neighbors.

Percolation theory mainly studies two percolation models: *Bond percolation* and *Site percolation* [4]. Let $G(V, E)$ be an infinite connected graph, where V is the set of nodes and E is the set of edges. In the bond percolation model on G, there is collection of $(X_e : e \in E)$ of independent Bernoulli random variables, each with the same mean pr, corresponding to the set E of edges (or 'bonds'). If $X_e = 1$, then the edge is *open*; otherwise it is *closed*. Given any two nodes x and y, x can reach y (i.e, $x \leftrightarrow y$), if there exists a path of open edges between x and y. The set of nodes, which can be reached by a specific node n_0 (e.g., the source of the broadcast) is denoted by C_0:

$$C_0 = \{x \in V : n_0 \leftrightarrow x\}. \quad (3)$$

The main interest is C_0 being infinite:

$$\theta^{bond}(p) = \mathbb{P}_p(|C_0| = \infty), \quad (4)$$

where $\mathbb{P}_p$ is the appropriate product probability measure on $\{0,1\}^E$. The *bond critical probability* $p_c^{bond}(G)$ is defined as:

$$p_c^{bond}(G) = sup\{pr : \theta^{bond}(pr) = 0\}, \quad (5)$$

so that $\theta^{bond}(pr) = 0$ if $pr < p_c^{bond}(G)$.

In the site percolation model, there is a collection of $(Y_x : x \in V)$ Bernoulli random variables, each with mean pr, corresponding to the set V of vertices. If $Y_x = 1$, then the node x is *active*, otherwise x is *inactive*. In this modes, a node y can reach a node x (i.e., $x \leftrightarrow y$) if there exists a path of consisting of active nodes only. Similar to definitions in the bond model, a site percolation probability $\theta^{site}(pr)$ and, a site critical probability $p_c^{site}(G)$ can be obtained. There is well-known inequality such that $p_c^{site}(G) \leq p_c^{bond}(G)$ and therefore, percolation does not occur in site model as readily as it does in bond model.

From the definitions of bond percolation and site percolation, we see that PBBF is a site percolation process.Specifically, a node x always sends a broadcast message either waiting for the next ATIM window or immediately. Therefore, we cannot apply bond percolation model to PBBF. On the other hand, the message may not be received by a node y, if the node is sleeping (inactive), which suggests a site percolation model. The probability a node may be sleeping depends on p and q parameters in PBBF. In particular, node y does not receive a broadcast when with probability p node x sends the broadcast without advertising and when in this case y may be sleeping with a probability of $1 - q$. Otherwise, y receives the broadcast (perfect time synchronization is assumed). Thereby, the probability that a broadcast dies at a node x in a given execution of PBBF:

$$p_{die}^{x} = p \times (1 - q)^{d(x)}. \quad (6)$$

However, a critical question is if x is a part of the disconnecting set N.

It may be possible to get an intuition of how $p_c^{site}(G)$ changes based on the network topology. For example, it is easy to observe that $0 < p_c^{site}(G) < 1$ (i.e., percolation exists), when G contains an edge, whose removal disconnects G [4]. Therefore, for PBBF the network topology should be at least 2-edge-connected so that percolation may not occur in the network. Essentially, connectivity and the number of paths in the network plays an important role in propagation of a broadcast. Recent studies using percolation theory [2, 13] show that in ad hoc networks there exists a critical node density,λ_c, below which the network is in a *subcritical phase* where nodes are partitioned into an infinite number of bounded clusters. Above λ_c, the network is in *supercritical phase* where there is one unbounded infinite cluster. In the case, where the node density, λ, is just above λ_c, it is shown that bottlenecks may appear. However, the node density has to be far above criticality to have a well-connected network with many paths from one node to another [2]. The analysis confirms that bottleneck nodes are of significant importance to efficient broadcast dissemination.