Adaptive Probability-Based Broadcast Forwarding in Energy-Saving Sensor Networks

CIGDEM SENGUL
INRIA-Saclay
INDRANIL GUPTA
University of Illinois at Urbana-Champaign
and
MATTHEW J. MILLER
Cisco Systems

Networking protocols for multihop wireless sensor networks (WSNs) are required to simultaneously minimize resource usage as well as optimize performance metrics such as latency and reliability. This article explores the energy-latency-reliability tradeoff for broadcast in WSNs by presenting a new protocol called *PBBF*. Essentially, for a given reliability level, energy and latency are found to be inversely related and our study quantifies this relationship at the reliability boundary. Therefore, PBBF offers an application designer considerable flexibility in the choice of desired operation points. Furthermore, we propose an extension to dynamically adjust the PBBF parameters to minimize the input required from the designer.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*; C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Performance

Additional Key Words and Phrases: Sensor network, broadcast, probabilistic protocols

This research was supported in part by NSF ITR grant CMS-0427089, in part by NSF CAREER grant CNS-0448246, and in part by an NDSEG fellowship.
A previous version of this article was, "Exploring the Energy-Latency Tradeoff for Broadcasts in Energy-Saving Sensor Networks," was presented at the 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005). ©IEEE 2005.
Authors' addresses: C. Sengul, Inria Saclay, Ile de France sud, Parc Club Orsay, Université ZAC des vignes, 4, rue Jacques Monod, 91893 Orsay Cedex, France; email: cigdem.sengul@inria.fr; M. J. Miller, Cisco Systems, Inc., 7025 Kit Creek Road, P.O. Box 14987, Research Triangle Park, NC 27709; email: matt@matthewjmiller.net; I. Gupta, University of Illinois, Urbana-Champaign, Department of Computer Science, Siebel Center for Computer Science, 201 N. Goodwin Avenue, University of Illinois, Urbana, IL 61801; email: indy@cs.uiuc.edu.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permission@acm.org.
© 2008 ACM 1550-4859/2008/03-ART6 $5.00 DOI = 10.1145/1340771.1340772 http://doi.acm.org/10.1145/1340771.1340772

ACM Reference Format:
Sengul, C., Miller, M. J., and Gupta, I. 2008. Adaptive probability-based broadcast forwarding in energy-saving sensor networks ACM Trans. Sens. Netw. 4, 2, Article 6 (March 2008), 32 pages. DOI = 10.1145/1340771.1340772 http://doi.acm.org/10.1145/1340771.1340772

1. INTRODUCTION

Sensor nodes are inherently resource constrained. For example, an off-the-shelf *Mote* (Crossbow Technology; go online to `http://www.xbow.com`.) has a lifetime of a few weeks (using a pair of standard AA batteries), short communication range distances, a 4-MHz processor, a few kilobytes of SRAM, and a few megabytes of Flash RAM. Offering better reliability and performance to a sensor network application (e.g., tracking, environmental observation) leads to greater usage and depletion of these resources. To support a wide variety of future applications, sensor networking technologies (hardware and software) will be required to provide enough flexibility for a designer to choose the appropriate operation point on the resource-performance spectrum.

In this article, we focus on the broadcast problem. Broadcast is useful to applications for disseminating sensor data, instructions, and code updates. Based on the specific application scenario, a broadcast protocol might be expected to satisfy different performance and resource usage requirements. For instance, for a query/response application, low latency might be critical for the operation of the network (e.g., less than 5 s in a 2- to 3-hop query area [Lu et al. 2005]). However, depending on the extent of data correlation, reliability can be traded off for the sake of energy consumption. On the other hand, a code update requires *all* nodes to be updated fairly quickly to maintain a consistent code image, and, hence, high reliability and medium latency become a priority for broadcast (e.g., 1–2 min for a 10-hop network [Levis et al. 2004]). For applications that can handle higher latency, energy consumption becomes a crucial factor determining network lifetime. Therefore, assuming an energy-saving sensor network, our goal is to design a broadcast protocol that allows a range of operating points from which an application designer can choose. To this end, we study a probabilistic approach to exploring the resource-performance tradeoff for broadcast communication.

While some previous studies of probabilistic broadcast in wireless networks exist outside the MAC protocol [Haas et al. 2002], we propose PBBF (Probability-Based Broadcast Forwarding), which works with the MAC protocol and can be integrated into any sleep scheduling mechanism. It must be noted that we do not propose a *new* MAC protocol in this article, but rather discuss a generic broadcast protocol that can be built into any MAC layer with an appropriate sleep scheduling strategy.

To address the energy constraints of battery-powered sensors, MAC protocols use a *sleep mode*, during which little power is consumed. Examples of such protocols include B-MAC [Polastre et al. 2004], T-MAC [van Dam and Langendoen 2003], S-MAC [Ye et al. 2002], and IEEE 802.11 PSM [IEEE 802.11

1999]. Based on the underlying *sleep scheduling mechanism*, at a given time, while some nodes are in *active mode*, others stay in sleep mode to save energy. PBBF can be added to such energy-saving MAC protocols via two new parameters: (1) p, which is the probability that a node rebroadcasts a packet *immediately* without ensuring that any of its neighbors are active, and (2) q, which is the probability that for a given node and a given time instant when it is supposed to be asleep based on its active-sleep schedule, the node instead stays awake in the expectation that it might be a receiver of an *immediate broadcast*.

Probabilistic broadcast schemes show threshold behavior; achieving a given level of reliability requires the probability of forwarding to be beyond a threshold. In Haas et al. [2002], this behavior was shown using the site percolation model. However, their approach, referred to as *Haas-Gossip* in the rest of this article does not allow an energy-latency tradeoff. Based on our analysis using the *bond percolation* model, we show that the two knobs, p and q, introduced by the PBBF protocol can be tuned to explore the energy-latency tradeoff. Essentially, only for some regions of values of p and q is the threshold condition for very high reliability satisfied, and we characterize the energy-latency tradeoff primarily in this region. We find that, in order to achieve a given application-defined level of reliability for broadcasts (i.e., fraction of nodes receiving the broadcast), the energy required and the latency obtained in PBBF are inversely related. While the inverse relation is not surprising, we precisely quantify the tradeoff, which is essential to delineate *tradeoff knobs* for the application designer. Based on these knobs, other techniques, such as propagating k most recent broadcasts with each packet, can be used in conjunction with PBBF to boost the reliability level without having a significant impact on energy or latency. While understanding how to set these tradeoff knobs is valuable, it is also desirable to operate PBBF adaptively with minimal support from the application designer. To this end, we propose an extension to PBBF, *adaptive PBBF*, which automatically configures p and q parameters to satisfy QoS requirements (i.e., energy, latency, and reliability levels) defined by the application designer.

In summary, the key contributions of this article are (1) a new probabilistic protocol, PBBF, for broadcasting, (2) a precise analysis of the energy-latency tradeoff allowed by PBBF for different levels of reliability, (3) fine-grained MAC-level simulation results of PBBF quantifying performance numbers for a typical broadcast application, (4) evaluation of PBBF with two sleep scheduling mechanisms, (5) evaluation of PBBF in comparison to the Haas-Gossip protocol, (6) adaptive PBBF, which adjusts tradeoff knobs based on QoS specification, and (7) simulation results of adaptive PBBF illustrating p and q convergence under different conditions (e.g., different network topologies and QoS requirements).

The rest of the article is organized as follows. Section 2 discusses energy-efficient communication in WSNs. In Section 3, we describe our proposed protocol, PBBF. An evaluation study of PBBF is presented in Section 4. Adaptive PBBF and its evaluation study are presented in Sections 5 and 6, respectively. Section 7 concludes the article and presents future directions of research.

2. ENERGY-EFFICIENT COMMUNICATION IN WIRELESS SENSOR NETWORKS

In this section, we discuss various approaches for energy-efficient data dissemination in wireless sensor networks. However, these approaches mostly work outside the MAC protocol. Therefore, we also present sleep scheduling mechanisms in wireless networks, which provide space for the design of an energy-efficient broadcast protocol in the MAC layer.

2.1 Efficient Broadcast Protocols

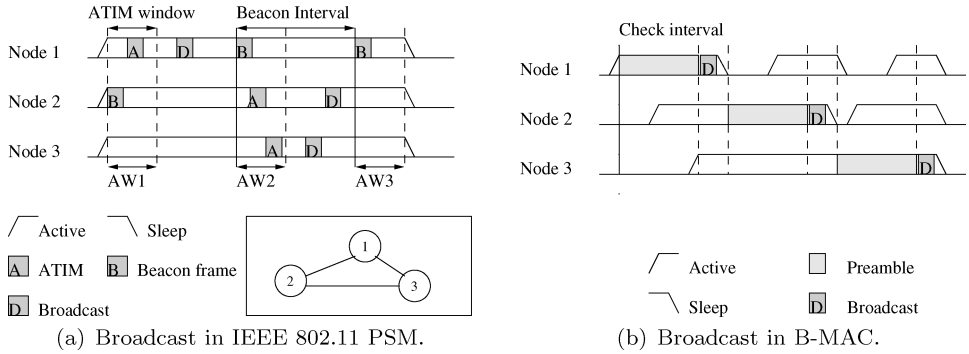
Broadcast is a fundamental communication primitive in sensor networks. Efficient broadcast techniques are essential for distributing software updates [Reijers and Langendoen 2003; Stathopoulos et al. 2003] or sensor observations [Heinzelman et al. 1999] among sensor nodes. The usual approach to broadcast is by flooding the entire network. This, however, creates a high number of redundant packets. While SPIN protocols [Heinzelman et al. 1999] incorporate *negotiation* in order to avoid deficiencies of the classic flooding approach, some approaches have explored the idea of overlaying a virtual infrastructure over the underlying network [Sivakumar et al. 1999; Sinha et al. 2001] to reduce the number of nodes involved in broadcasts. Finally, the problems with flooding can also be alleviated allowing each node to forward a message with some probability (i.e., gossip) [Haas et al. 2002; Sasson et al. 2003]. Our work in this article is most similar to this type of approach.

It has been shown that Haas-Gossip [Haas et al. 2002] exhibits bimodal behavior: either virtually all or virtually none of the nodes receive the broadcast based on the gossiping probability. This problem has been well studied in percolation theory, which studies the existence of a threshold value below which infinitely many finite clusters exist and above which the cluster size approaches infinity significantly quickly [Grimmet and Stacey 1998]. Similarly to Haas-Gossip, PBBF also affects the number of nodes that receive a broadcast since the broadcast may propagate when some nodes are in sleep mode. However, while Haas-Gossip is a *site percolation problem*, where nodes broadcast with some probability [Grimmet and Stacey 1998], PBBF corresponds to a *bond percolation problem*, where bonds are open (i.e., a broadcast is sent and received) with some probability. By changing the probability a link exists in the network; PBBF provides the ability to tune the performance of an application based on the tradeoff between energy, latency, and reliability.

2.2 Sleep Scheduling Mechanisms

There are two main MAC-layer approaches to reduce energy consumption in WSNs. The first approach is to use an active-sleep cycle, which lets nodes sleep periodically. The second approach involves using an additional low-power *wake-up radio* to wakeup nodes [Schurgers et al. 2002]. However, since this approach requires an extra hardware component on the sensor node, the remainder of the article focuses on only the active-sleep cycle approach.

The basic idea of introducing an active-sleep cycle to a contention-based protocol is to divide time into frames. Each frame is divided into an *active time*



(a) Broadcast in IEEE 802.11 PSM. (b) Broadcast in B-MAC.

Fig. 1. Broadcast with different sleep scheduling mechanisms.

and a *sleep time*. During the sleep time, a node puts its radio in sleep mode to save energy. During the active time, a node can send and receive messages. For instance, the IEEE 802.11 protocol [IEEE 802.11 1999] provides such a power-save mode (PSM), which requires nodes to be time synchronized and follow the same active-sleep schedule. S-MAC [Ye et al. 2002] proposes *virtual clustering* of neighbors to autosynchronize active-sleep schedules. In both IEEE 802.11 PSM [IEEE 802.11 1999] and S-MAC [Ye et al. 2002], active and sleep times are fixed, while in T-MAC [van Dam and Langendoen 2003] nodes dynamically determine the length of active times based on communication rates. B-MAC [Polastre et al. 2004] allows more flexible duty cycles by using preamble sampling to detect channel activity. If activity is detected, a node stays awake and returns to sleep after reception. Otherwise, a node switches to sleep after preamble sampling.

Since the focus of this article is broadcast, we next discuss the behavior of these sleep scheduling mechanisms for this communication pattern. Figure 1(a) shows an example for IEEE 802.11 PSM, where nodes are synchronized to wake up at the beginning of every *beacon interval*. Pending traffic is announced via ATIMs (Ad-hoc Traffic Indication Messages) in an ATIM window. In the example, Node 1 announces a broadcast ATIM for which all one-hop nodes (or *neighbors*) (e.g., Nodes 2 and 3) should stay awake to receive the message after the ATIM window. An immediate observation is that to rebroadcast the message, a node must wait for the next ATIM window to guarantee that each neighbor receives the ATIM advertising the broadcast. This increases latency. A second observation is that when, say, Node 2 retransmits the broadcast message, Nodes 1 and 3 receive redundant packets. Furthermore, due to redundant broadcast packets, nodes stay awake the entire beacon interval more often, mostly listening on the channel. This increases energy consumption.

While S-MAC would exhibit similar latency performance, the energy consumption of broadcast is somewhat different. In S-MAC, nodes stay awake for fixed intervals, called *listen intervals*, and traffic is sent in each interval without advertisements. Hence, broadcast traffic does not increase the energy spent in idling; however, energy consumption still increases due to redundancy. Additionally, nodes that follow more than one schedule add to redundancy since

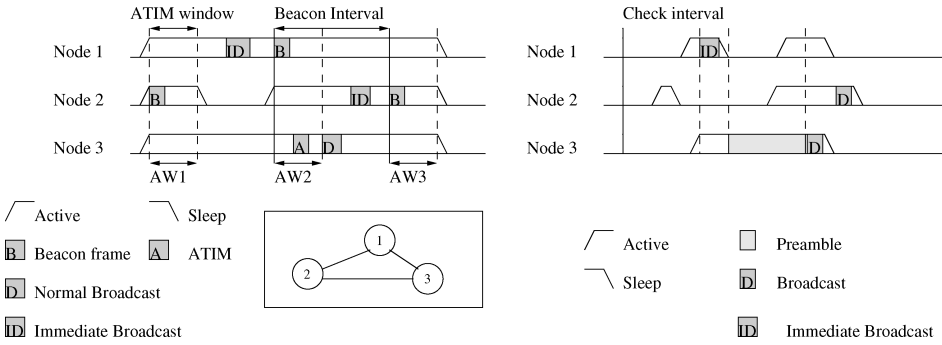
these nodes typically transmit a broadcast message multiple times to guarantee the neighbors with different schedules receive the message. The global schedule algorithm [Li et al. 2005] addresses this problem by allowing all nodes eventually to converge to the same schedule. However, the broadcast redundancy problem remains as all nodes send the message once.

While the same observations can be made for T-MAC, B-MAC is more similar to IEEE 802.11 PSM. In B-MAC nodes wake up every *check interval* to listen for activity. For each broadcast packet, nodes need to send and receive a preamble at least as long as the check interval. Hence, the energy spent for listening increases with redundant packets. Consider the example shown in Figure 1(b). Node 1 sends a long preamble before it sends the broadcast message. Nodes 2 and 3 wake up asynchronously and remain awake as they hear the preamble. Two factors incur high energy consumption in B-MAC: (1) depending on when they wake up, nodes need to remain on until they hear the actual packet, and (2) each packet is preceded by a long preamble. Additionally, the length of the preamble affects latency. While recently SCP [Ye et al. 2006] was proposed to reduce the costs associated with preamble listening, redundancy inherent in broadcast communication still affects energy consumption. Therefore, these sleep scheduling mechanisms for sensor networks display similar disadvantages in the presence of broadcast traffic. Motivated by these observations, we propose Probability-Based Broadcast Forwarding (PBBF), which allows tradeoffs for latency, energy consumption, and reliability.

3. PROBABILITY-BASED BROADCAST FORWARDING

We propose using Probability-Based Broadcast Forwarding, which can be used in conjunction with any sleep scheduling mechanism. PBBF exploits the redundancy in broadcast communication and forwards packets using a probability-based approach. PBBF introduces two new parameters to a sleep scheduling protocol: p and q. The first parameter, p, is the probability that a node rebroadcasts a packet in the current active time despite the fact that not all neighbors may be awake to receive the broadcast. The second parameter, q, represents the probability that a node remains on after the active time when it normally would sleep.

Figure 2(a) shows a simple example of PBBF integrated into IEEE 802.11 PSM. In the example, Node 1 has a broadcast message to send after $AW1$. Using the p parameter, Node 1 decides to send the message immediately instead of waiting for $AW2$ to announce it. Therefore, only Node 3, which tossed a coin and decided to stay awake after $AW1$ based on the q parameter, receives the message. On reception of the message, Node 3 decides to rebroadcast via a normal broadcast and, therefore, waits for $AW2$ to guarantee that each node in its neighborhood receives the broadcast. Hence, Node 2 is able to receive the message this time. This example shows that, if a node chooses to rebroadcast immediately, only the subset of neighbors that are currently awake can receive the packet, but with no sleep-induced delay. However, there may be no nodes to receive the packet (e.g., this would be the case if Node 3 were not awake after $AW1$ when Node 1 transmitted). The q parameter is used to avoid this



(a) Broadcast in IEEE 802.11 PBBF/PSM. (b) Broadcast in PBBF/BMAC.

Fig. 2. Broadcast in PBBF with different sleep scheduling mechanisms.

problem as much as possible by allowing nodes to stay awake regardless of their active-sleep schedules.

Next, we discuss how PBBF changes the operation of B-MAC in the same network (see Figure 2(b)). In this example, Node 1 decides to send an immediate broadcast, and therefore, sends the message without a long preamble. Node 3, which tossed a coin and decided to stay awake, receives the message. Next, Node 3 decides to rebroadcast via a normal broadcast and, hence transmits a long preamble preceding the message. Hence, Node 2 is able to detect channel activity and receive the message this time.

Figure 3 shows pseudocode of changes to any sleep scheduling protocol required for PBBF. The original sleep scheduling protocol is a special case of PBBF with $p = 0$ and $q = 0$. Essentially, through p and q, PBBF determines how closely the nodes should follow the underlying sleep scheduling protocol. The *always-on* mode (i.e., no active-sleep cycles) can be approximated by setting $p = 1$ and $q = 1$. PBBF is still slightly different than *always-on* in this case because it still has the byte overhead (e.g., sending synchronization beacons) and temporal overhead (e.g., PBBF cannot send data packets during the ATIM window) of active-sleep cycles.

Through the use of two parameters, p and q, the PBBF protocol provides a tradeoff between energy, latency, and reliability. While p presents a tradeoff between latency and reliability (i.e., the fraction of nodes receiving a broadcast), q presents a tradeoff in terms of energy and reliability. As p increases, latency decreases while the fraction of nodes not receiving a broadcast increases (unless $q = 1$). As q increases, energy consumption increases, but the fraction of nodes receiving a broadcast increases (unless $p = 0$).[1] By specifying these two parameters, we investigate the energy, latency, and reliability tradeoffs in the next section.

[1]It must be noted that energy consumption is affected differently for different sleep scheduling mechanisms. For instance, compared to S-MAC and T-MAC, PBBF is expected to provide higher energy savings for IEEE 802.11 PSM and B-MAC as it can also decrease idle listening.

```
SLEEP-DECISION-HANDLER()
1  /* Called at the end of active time */
2  /* If stayOn is true, remain on; otherwise sleep*/
3  stayOn ← false
4
5  if DataToSend = true or DataToRecv = true
6    then
7         stayOn ← true
8    else if UNIFORM-RAND(0, 1) < q
9           then stayOn ← true


RECEIVE-BROADCAST(pkt)
1  /* Called when broadcast packet pkt is received */
2  if UNIFORM-RAND(0, 1) < p
3    then SEND(pkt)
4    else ENQUEUE(nextPktQueue, pkt)
```

Fig. 3. Pseudocode for PBBF.

4. EVALUATION OF PBBF

The goal of our performance study is to evaluate PBBF in terms of its ability to tune latency, energy, and reliability of broadcast. Furthermore, in Sections 4.1 and 4.2, we validate that PBBF is not specific to a sleep scheduling mechanism by studying PBBF in conjunction with IEEE 802.11 PSM [IEEE 802.11 1999] and B-MAC [Polastre et al. 2004]. We use PBBF/* when referring to PBBF with respect to a specific sleep scheduling mechanism, where * is either IEEE-802.11 PSM or B-MAC. Section 4.3 evaluates PBBF in comparison to the Haas-Gossip protocol [Haas et al. 2002]. In our study, we first assume ideal MAC and physical layers but relax this assumption in Section 4.4 and evaluate PBBF in a more realistic environment.

4.1 Analytical Results

We analyze PBBF by using a combination of theory and simulations. Simulations are required because we find a complete analysis to be intractable, in spite of several available theoretical frameworks such as percolation theory. For the simulations used in this section, we use IEEE 802.11 PSM as the sleep scheduling protocol.

We consider a grid network topology, where each node is connected to four neighbors except the nodes on the boundary (i.e., a square lattice with no wrapping on the axes) and the broadcast source is as near to the center of the grid as possible. Table I lists the parameters used in the simulation part of the analysis. N is the number of nodes, λ is the rate that broadcasts are generated at the source, and T_{active} and T_{frame} are the times nodes are active each frame and the time between frames, respectively. L_1 is a latency value described in Section 4.1.3. Its chosen value is based on empirical data observed in our simulations in Section 4.4. We use P_I as the power level when a node is active and

Table I. Analysis Parameter Values

Parameter	Value
N	5625 (75×75)
P_{TX}	81 mW
P_I	30 mW
P_S	3 μW
λ	0.01 packets/s
L_1	$\approx$ 267 ms
T_{frame}	10 s
T_{active}	1 s

P_S as the power level when a node is sleeping. The values we use are based on Mica2 Motes (Crossbow Technology; go online to `http://www.xbow.com`.)

4.1.1 *Reliability.* The reliability of PBBF can be analyzed using *percolation* models. Percolation theory states that a gossip initiated by a source, n_0 dies out if there is a set of nodes, $\mathbb{D}$, that disconnects n_0 from the rest of the graph. In PBBF, $\mathbb{D}$ is the set of nodes that send an immediate broadcast which is not received by any of its neighbors.

Percolation theory mainly studies two percolation models: *bond percolation* and *site percolation* [Grimmet and Stacey 1998]. 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, p_{edge}, 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, where

$$C_0 = \{x \in V : n_0 \leftrightarrow x\}. \tag{1}$$

Percolation theory calculates conditions under which C_0 is infinite, in other words, the values of p_{edge} for which the probability $\theta^{bond}(p_{edge})$ of the component C_0 being of infinite size is close to 1.

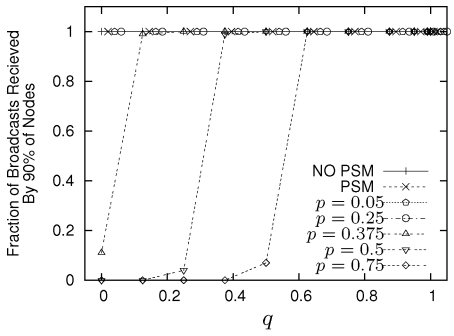
The *bond critical probability*, $p_c^{bond}(G)$, is defined as

$$p_c^{bond}(G) = sup\{p_{edge} : \theta^{bond}(p_{edge}) = 0\}, \tag{2}$$

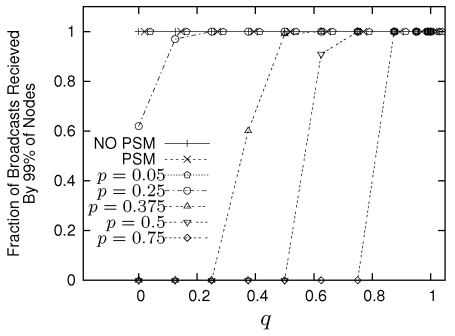
so that $\theta^{bond}(p_{edge}) = 0$ if $p_{edge} < p_c^{bond}(G)$.

The site percolation model differs because, instead of cutting given edges (bonds) in the graph with some probability, each node (site) in the graph is subjected to removal with some probability. This corresponds to the analysis of the Haas-Gossip protocal [Haas et al. 2002], where each node decides probabilistically whether to broadcast to either all its neighbors or none of them.

PBBF's reliability is characterized by a bond percolation model. First, if a node A receives the broadcast message, the probability that a given neighbor, B, of A receives a copy of the message via the link $A \rightarrow B$ is $p \cdot q + (1 - p)$. The first term arises from the likelihood of A broadcasting the message immediately after reception *and* that B being awake at the time. The second term is simply the likelihood of a rebroadcast when B is awake (i.e., the beginning



(a) 90% reliability. PSM and $p = \leq 0.25$ lines are perturbed for the sake of clarity.



(b) 99% reliability. Lines for PSM and $p = 0.05$ are perturbed for the sake of clarity.

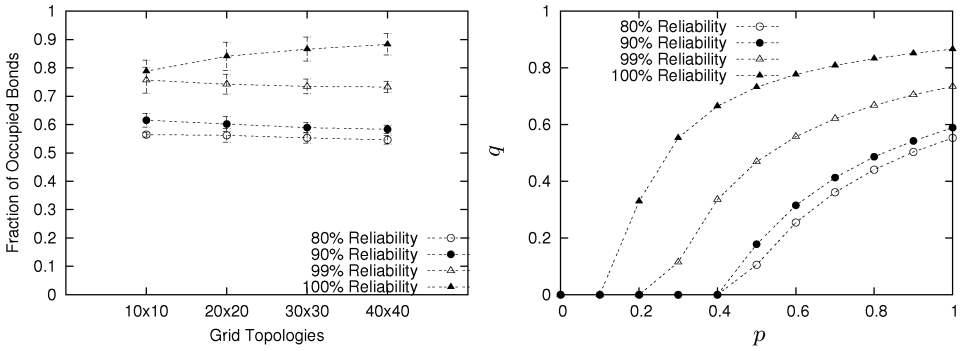
Fig. 4. Reliability of NO-PSM, PSM = IEEE 802.11 PSM, and the threshold behavior of PBBF/802.11 PSM ($p = *$ lines).

of next active time). Then each (directed) edge in the network is *open* with this probability. It must be noted that, even though we assume symmetric links, a broadcast traverses a link only once, since nodes drop a broadcast packet if they receive a duplicate. Hence, by associating each (directed) edge in the network with a probability $p_{edge} = 1 - p \cdot (1 - q)$ of being present, we can say the following [Grimmet and Stacey 1998]:

Remark 4.1 (*p and q for High Reliability*). If $p_{edge} = 1 - p \cdot (1 - q) \geq p_c^{bond}(G)$, the broadcast is received at infinitely many nodes.

We next show reliability of PBBF/IEEE 802.11 PSM by varying q while keeping p fixed. For each level of reliability (e.g., 90% and 99%), threshold behavior is observed as shown in Figures 4(a) and 4(b). For sufficiently large values of p, none of the broadcasts achieve the desired reliability when q is small. However, at some threshold q value, reliability rapidly improves to where every broadcast is received by the specified fraction of nodes. For instance, for $p \leq 0.25$, the fraction of broadcasts received by 90% of the nodes is 1 (these lines overlap with the PSM and NO-PSM lines). On the other hand, for 90% reliability, $p \geq 0.375$, and for 99% reliability, $p \geq 0.25$, result in a threshold behavior. This is similar to the critical probability behavior shown in percolation theory [Grimmet and Stacey 1998].

We use a fast Monte Carlo algorithm from [Newman and Ziff 2001] to investigate the critical bond ratio for different reliability measures in grid networks (see Figure 5(a)). For a higher level of reliability, as expected, a larger number of bonds is required to be present. The fraction of occupied bonds shows only slight variations as the network size increases. These variations increase for higher reliability due to boundary effects. The p and q values necessary to achieve various levels of reliability in 30×30 grid network are shown in Figure 5(b). Each point in the figure represents p and q values to achieve the p_c^{bond} for a 30×30 grid network. Essentially, these results show the direct relationship between p



(a) p_c^{bond} for various grid sizes.

(b) Relationship between p and q for a given reliability level in a 30×30 grid network.

Fig. 5. Relationship of p and q in grid networks.

and q for a given level of reliability. For instance, the line for 100% reliability crosses x-axis at $p = 0.1$. Therefore, while below $p = 0.1$, $q = 0$ satisfies 100% reliability, above $p = 0.1$, q should be chosen from the region above 100% line. However, it must be noted that for the studied case (i.e., when a single message is broadcast in the network), $p > 0$ and $q = 0$ corresponds to pure gossiping and might lead to wasted energy. On the other hand, when multiple messages broadcast in the network, setting $p > 0$ and $q = 0$ allows sending immediate broadcasts to nodes that are already awake to receive another message. Hence, the awake times of nodes are more effectively utilized.

As expected, for a lower reliability level, q can stay 0 for higher p values. For instance, for 90% reliability, $q = 0$ as long as $p \leq 0.4$. Using Figures 5(a) and (b), we can see that $p_c^{bond}(G) \approx 0.6$ achieves 90% reliability in a grid, and when $p = 0.5$ and $q \geq 0.18$, $p_{edge} = 1 - p \cdot (1 - q) \geq 0.6$ can be satisfied. This can be verified by looking at $p = 0.5$ in Figure 4(a), where a threshold behavior is observed when $q \approx 0.25$. Therefore, for a given p, q should be selected from the region above the line corresponding to a reliability level.

4.1.2 *Energy.* Assuming the underlying sleep scheduling protocol divides time into frames and denoting active time as T_{active} and sleep time as T_{sleep}, relative energy consumption of a sleep scheduling protocol compared to a protocol with no energy-saving, $E_{original}$, can be written as

$$E_{original} = \frac{T_{active}}{T_{frame}}, \tag{3}$$

where $T_{frame} = T_{active} + T_{sleep}$. The PBBF protocol allows nodes to stay active, regardless of their active-sleep schedules, based on the q parameter. Therefore, the new active and sleep times in PBBF, $T_{active:PBBF}$ and $T_{sleep:PBBF}$, are

$$T_{active:PBBF} = T_{active} + q \cdot T_{sleep}, \tag{4}$$

$$T_{sleep:PBBF} = (1 - q) \cdot T_{sleep}. \tag{5}$$

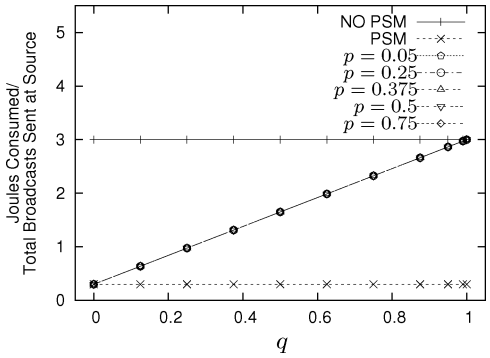


Fig. 6. Average energy consumption of NO-PSM, PSM = IEEE 802.11 PSM, and PBBF/IEEE 802.11 PSM ($p = *$ lines). All the PBBF lines overlap.

The relative energy consumption of PBBF, E_{PBBF}, is

$$E_{PBBF} = \frac{T_{active:PBBF}}{T_{frame}} = \frac{T_{active} + q \cdot T_{sleep}}{T_{frame}}. \tag{6}$$

The increased energy consumption due to the q parameter compared to original sleep scheduling protocol is

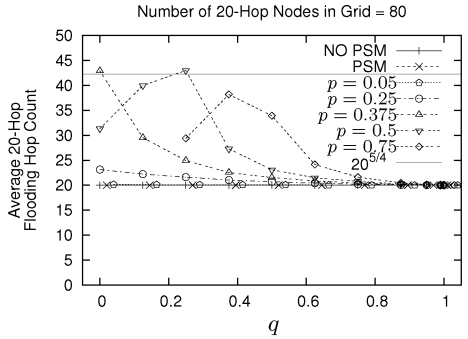
$$\frac{E_{PBBF}}{E_{original}} = \frac{T_{active} + q \cdot T_{sleep}}{T_{active}} = 1 + q \cdot \frac{T_{sleep}}{T_{active}}. \tag{7}$$

Although T_{active} and T_{sleep} are assumed to be fixed in Equation (7), these parameters can also be variables of a probabilistic distribution. The simulation results verify the analytical result given in Equation (7) (see Figure 6).[2] While using PSM saves almost 3 J/broadcast over using no PSM, the energy consumption of PBBF increases linearly with the q parameter, and does not depend on p at all (the lines for different values of p overlap).

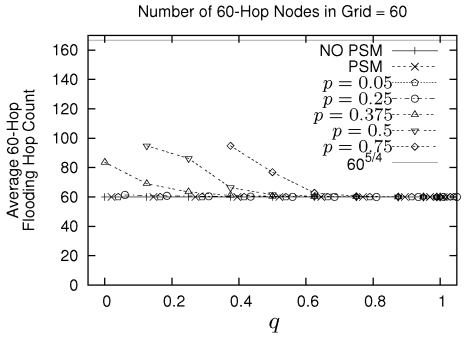
4.1.3 *Latency.* For a given node, A, and a neighbor of A, B, we calculate the expected time, L, between A sending the broadcast and B receiving it from A (assuming a successful transmission from A to B). The probability that the broadcast is sent and received immediately is $p \cdot q$, the product of the probability of an immediate broadcast (p) and that node B stays awake (q). The probability of the broadcast being sent with the assurance that all nodes wake up is simply $(1 - p)$. Thus, if the time to immediately transmit the data packet is denoted as L_1 and the time to wake up all neighbors for the broadcast is L_2, then L can be calculated as:

$$\begin{aligned} L &= \frac{L_1 \cdot p \cdot q + (L_1 + L_2) \cdot (1-p)}{p \cdot q + (1-p)} \\ &= L_1 + L_2 \cdot \frac{1-p}{1-p+p \cdot q}. \end{aligned} \tag{8}$$

[2]More precisely, $\frac{E_{PBBF}}{E_{original}} = 1 - (1 - p_{edge}) \cdot \frac{T_{rx}}{T_{active}} - (1 - (1-p)(1-q)) \cdot \frac{T_{idle}}{T_{active}} + q \cdot \frac{T_{sleep}}{T_{active}}$, where T_{rx} and $Tidle$ are the time spent in reception and idling originally. However, although PBBF reduces T_{rx} and T_{idle}, since $\frac{T_{rx}}{T_{active}} \leq 1$, $\frac{T_{idle}}{T_{active}} \leq 1$ and $\frac{T_{sleep}}{T_{active}} \gg 1$, Equation (7) holds.



(a) 20 hops. Lines for PSM and $p = 0.05$ are perturbed for the sake of clarity.



(b) 60 hops. Lines for PSM and $p \leq 0.25$ are perturbed for the sake of clarity.

Fig. 7. Average hops traveled by a broadcast to reach a node (a) 20 hops and (b) 60 hops from the source for NO-PSM, PSM = IEEE 802.11 PSM, and PBBF/IEEE-802.11 PSM ($p = *$ lines).

It must be noted while L_1 is determined by the MAC protocol (i.e., the channel access time), L_2 depends on how the sleep scheduling mechanism handles broadcast communication. Essentially, L_2 is determined by when a node A receives the broadcast during T_{frame} and how long it takes to ensure all neighbors receive the broadcast packet. L_1 and L_2 can be either constants or variables of a probabilistic distribution. In our study with IEEE 802.11 PSM, we observed $L_1 \approx 267$ ms. Furthermore, in our simulations, nodes typically received advertised broadcast packets at the end of an ATIM window; hence, $L_2 \approx T_{frame} = 10$ s (see Table I).

When calculating the overall latency, we need to account for the fact that a broadcast can potentially traverse through multiple different paths from the source node S to a given node B. In other words, the actual latency from S to B is a function of L and the average hop count, $hop(S, B)$, from S to B:

$$L_{S,B} = L \cdot hop(S, B). \tag{9}$$

$hop(S, B)$ may be greater than the hop count of shortest path from S to B since links exist on the graph based on p_{edge}. In a grid network, when the source broadcasts a packet, the packet starts propagating in four directions. Since nodes that receive a duplicate do not rebroadcast, each broadcast message builds a uniform spanning tree. It has been shown that on such a spanning tree, the expected number of vertices on the arc from the source that lie within a hop distance d is $d^{5/4+o(1)}$ [Kenyon 1999; Guttmann and Bursill 1990]. From this, we can upper bound the average latency of a broadcast to reach a node B at a hop distance d from S as follows:

$$L_{S,B} \leq L \cdot d^{5/4+o(1)}, \tag{10}$$

where d is the hop distance between S and B. From Figures 7(a) and (b), we observe that the latency $L_{S,B}$ is indeed proportional to d as the reliability approaches to 100% (points toward the right-hand side of the plots). Essentially, as reliability increases, broadcast packets traverse direct paths and, hence, nodes that are 20 (60) hops away from the source receive the broadcast in

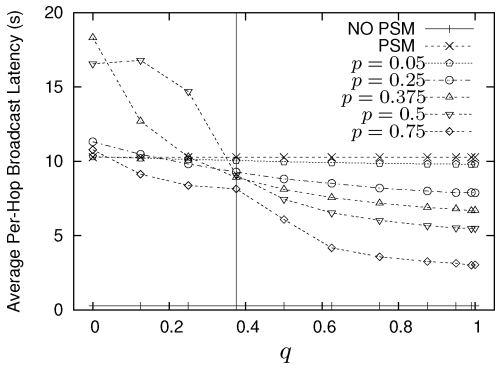


Fig. 8. Average per-hop broadcast latency of NO-PSM, PSM = IEEE 802.11 PSM, and PBBF/IEEE-802.11 PSM ($p = *$ lines) . The line at $q = 0.375$ shows when PBBF starts behaving as expected: higher p values result in lower latency.

20 (60) hops. However, as reliability decreases, nodes receive packets through longer paths and the latency is within the bound in Equation (10).

The variation of per-hop latency versus q is shown in Figure 8. Since only nodes that receive at least one broadcast are included in this latency calculation, at small values of q, the lower latency achieved by lower p values is misleading. However, as q increases (i.e., broadcasts reach more nodes), higher p values (e.g., $p = 0.5$) achieve lower latency as nodes do not incur wakeup latency (i.e., L_2).

4.1.4 *Energy-Latency Tradeoff.* From Equations (7) and (8), we can derive the direct relation between energy, E_{PBBF}, and latency, L, as

$$E_{PBBF} = \left(1 - \frac{L_2 + L_1 - L}{L - L_1} \cdot \frac{1-p}{p} \cdot \frac{T_{sleep}}{T_{active}}\right) \cdot E_{original}. \tag{11}$$

Equation (7) shows that the energy consumed at a node increases linearly with q. Equation (8) shows that the latency is inversely related to q (and also p). Thus, the energy and latency are inversely related to each other in PBBF. Determining the minimum value of q for a given value of p that gives 99% reliability (see Figure 4(b)), the energy-latency trade-off with 99% reliability is illustrated in Figure 9.

In summary, the threshold behavior of PBBF allows an application designer to first set p and q so that they are just across the reliability threshold boundary and into the high reliability region. Second, these values can be further tuned (staying close to the boundary) until the desired energy-latency tradeoff is achieved.

4.2 PBBF with a Sensor MAC Protocol

Since IEEE 802.11 PSM might not be a good match for sensor networks, in this section, we study the performance of PBBF with a sensor MAC protocol, B-MAC. We again assume an ideal MAC and physical layer with no collisions or interference. The parameters used in this study are the same as the parameters used in Section 4.1 (see Table I) with the following exceptions. In

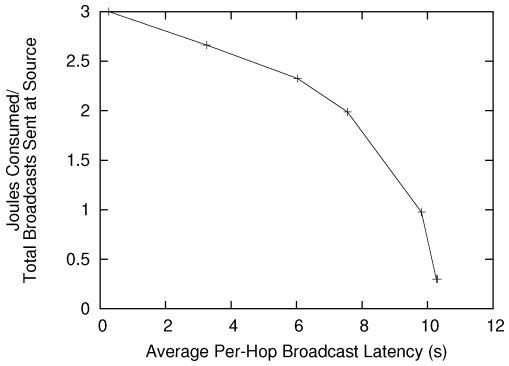
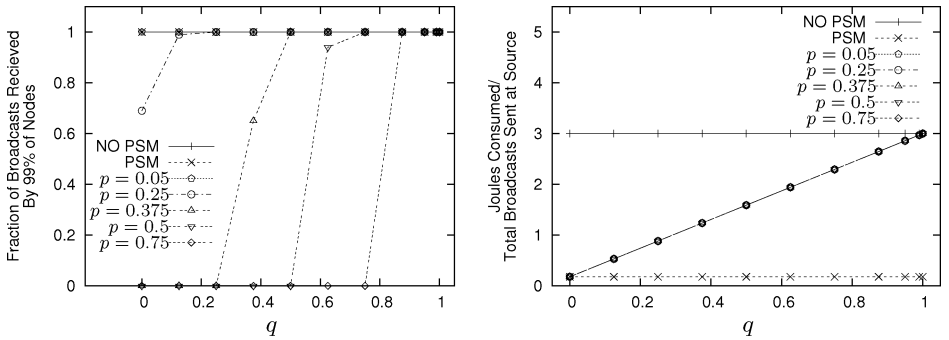


Fig. 9. Energy-latency tradeoff for 99% reliability for PBBF/IEEE 802.11 PSM.



(a) Threshold behavior for 99% reliability.

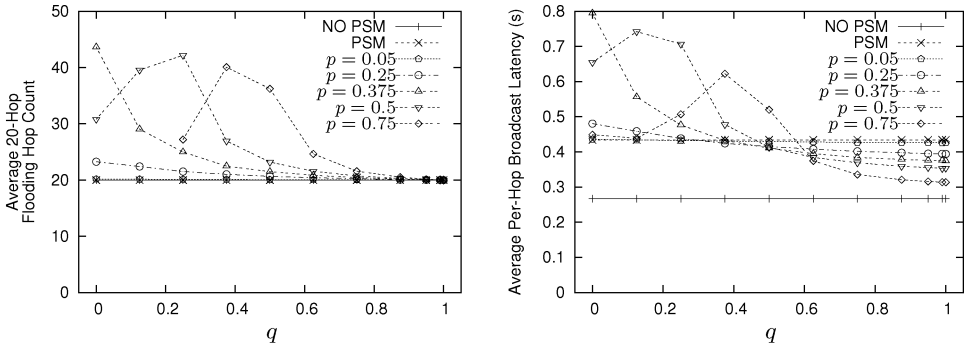
(b) Average energy consumption. All PBBF lines overlap.

Fig. 10. Reliability and energy performance of NO-PSM, PSM = B-MAC, and PBBF/B-MAC ($p = *$ lines).

B-MAC, $T_{frame} = 0.135$ s and $T_{active} = 0.008$ s. Furthermore, using a preamble of 371 bytes, nodes keep the channel busy approximately 0.15 s to guarantee all neighbors are awake before each data transmission. Hence, while $L_1 \approx 0.267$ s, $L_2 \approx= 0.15$ s.

As in Section 4.1, we evaluate PBBF in terms of reliability, energy consumption and latency. Simulation results show that PBBF/B-MAC exhibits similar trends with PBBF/IEEE 802.11 PSM (see Section 4.1). PBBF/B-MAC achieves slightly higher reliability (3–10% higher) at the transition point (i.e., the point when the reliability starts improving). For instance, when $p = 0.25$ and $q = 0$, the fraction of broadcasts received by 99% of the nodes with PBBF/B-MAC is 69%, while with PBBF/IEEE 802.11 PSM, it is 62%. However, the transition point is the same for both (see Figures 4(b) and 10(a)). This is expected as the threshold behavior is mainly determined by p and q parameters and is independent of the specifics of the underlying sleep scheduling mechanism.

In terms of energy consumption, B-MAC is able to maintain 0.178 J/broadcast (see Figure 10(b)), while IEEE 802.11 PSM achieves 0.3 J/broadcast (see



(a) Average hops traveled by a broadcast to reach a node at 20 hops.

(b) Average per-hop broadcast latency.

Fig. 11. Latency performance of NO-PSM, PSM = B-MAC, and PBBF/B-MAC ($p = *$ lines).

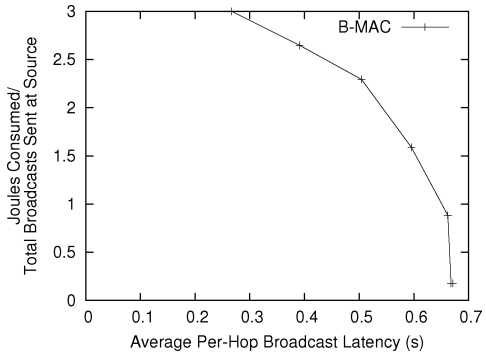
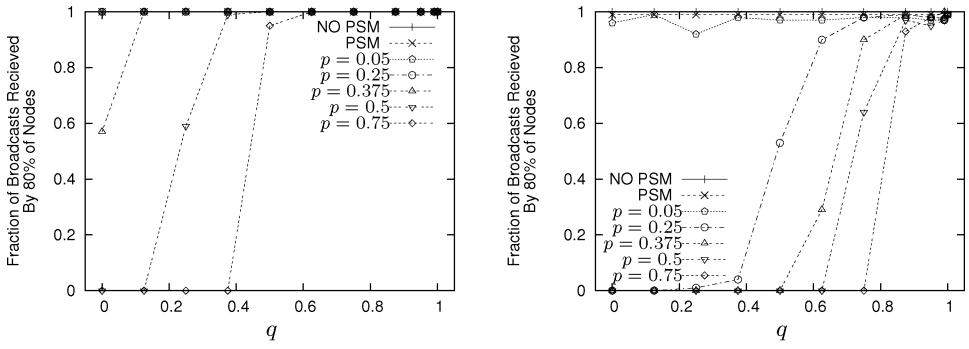


Fig. 12. Energy-latency tradeoff for 99% reliability for PBBF/B-MAC.

Figure 6). Hence, B-MAC improves energy consumption by 68% over IEEE 802.11 PSM. However, regardless of this difference, PBBF is able to span the interval defined by the minimum (i.e., pure PSM) and the maximum (i.e., no PSM) energy consumption.

Figure 11(a) depicts the latency performance in terms of the average number of hops traveled by a broadcast to reach a node at 20 hops. We observe that PBBF/B-MAC and PBBF/IEEE 802.11 PSM show negligible variation in performance (less than 0.2%) since the number of hops a broadcast travels is mainly dependent on the p and q parameters (and the reliability level achieved by these parameters) (see Figures 7(a) and 11(b). However, in terms of per-hop latency, lower latency is attained compared to PBBF/IEEE 802.11 PSM (see Figures 8 and 11(b)). Essentially, per-hop latency is primarily determined by T_{frame}, which is 0.135 s in B-MAC and 10 s in IEEE 802.11 PSM. However, the point where higher p values start performing with lower latency is $q = 0.375$, as in PBBF/IEEE 802.11 PSM. Furthermore, as we see in Figure 12, PBBF/B-MAC and PBBF/IEEE 802.11 PSM have similar energy-latency tradeoff. We observe that for both PBBF/IEEE 802.11 PSM and PBBF/B-MAC the energy-latency curve is concave down and decreasing.



(a) No Haas-Gossip. (b) With Haas-Gossip.

Fig. 13. The impact of Haas-Gossip on NO-PSM, PSM = IEEE 802.11 PSM, and the threshold behavior of PBBF/IEEE 802.11 PSM for 80% reliability.

In summary, these results show that PBBF is not limited to IEEE 802.11 PSM and can provide the ability to tune energy, latency, and reliability with different sleep scheduling mechanisms. Essentially, PBBF is able to span the energy-latency-reliability region, which is defined by the specific sleep scheduling mechanism, by varying its p and q parameters.

4.3 PBBF versus Haas-Gossip

PBBF is designed with the goal of providing application designers some knobs to control the quality of broadcast communication in energy-saving sensor networks. In this section, we emphasize PBBF's ability to tune reliability, energy, and latency by comparing its performance with Haas-Gossip protocol [Haas et al. 2002]. The underlying sleep scheduling mechanism is IEEE 802.11 PSM. The parameters used in this study are the same as the parameters used in Section 4.1.

In Haas-Gossip protocol, a node advertises a broadcast in an ATIM window with a gossiping probability gp, or drops the packet. We first set the gossiping probability $gp = 0.7$ based on [Haas et al. 2002] and evaluate reliability of the protocols (including PBBF) with Haas-Gossip. Figure 13(b) shows that, compared to Figure 13(a), in all protocols, reliability is adversely affected. This is expected since Haas-Gossip cuts all the edges of a node, while PBBF cuts a subset of these edges based on p and q parameters. Figure 14 shows the energy-latency trade-off of Haas-Gossip when gp is varied between 0.7–1.0. For all gp values, the fraction of nodes that receive *at least* 90% of the broadcast packets is 99%.[3] Increasing gp improves latency; however, since no broadcast is sent immediately, unlike PBBF, the improvement is lower-bounded by the latency that can be achieved by the underlying sleep scheduling mechanism ($\approx 10s$ in IEEE 802.11 PSM). Furthermore, Haas-Gossip does not provide much opportunity for tuning energy consumption. This is because Haas-Gossip does not affect the

[3]It must be noted that the energy-latency tradeoff graph for Haas-Gossip is plotted differently than PBBF. For PBBF, energy-latency tradeoff is plotted for a fixed level of reliability (e.g., 99%), whereas Haas-Gossip achieves higher reliability with higher gp (e.g., *at least 90%*).

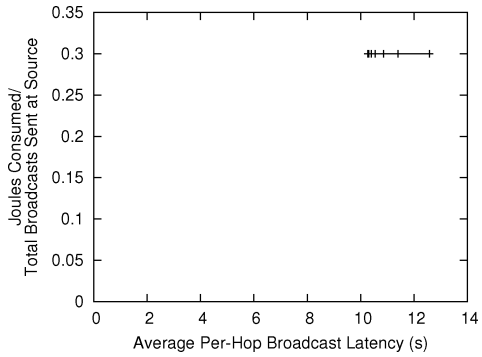


Fig. 14. Energy-latency tradeoff for Haas-Gossip for at least 90% reliability.

sleep scheduling of nodes, but saves energy from reducing the number of transmissions and receptions. Since we take $P_I = 30$ mW as the active energy cost, we only take into account the cost of receptions ($P_{RX} = P_I$ (Crossbow Technology; go online to `http://www.xbow.com`.)), but not transmissions. However, the energy to transmit a broadcast constitutes 4.5% of average energy use per broadcast. Hence, we can conclude that energy-latency tradeoff of Haas-Gossip does not provide tuning capability as PBBF does.

While PBBF might not be the optimal solution to reduce latency and energy consumption, the strength of PBBF lies in its ability tune performance locally. For instance, an approach that reserves the channel k-hops ahead and transmits the broadcast in one beacon interval can provide both energy and latency savings. However, k-hop channel reservation requires careful selection of nodes that should transmit the channel reservation messages in the k-hop neighborhood of the broadcast sender (e.g., might need to find the connected dominating set in the k-hop neighborhood of the broadcast sender to optimize performance). Therefore, the advantage of PBBF becomes obvious when we consider the complexity of such an approach in comparison to PBBF, where each node makes its decisions locally and independently of other nodes.

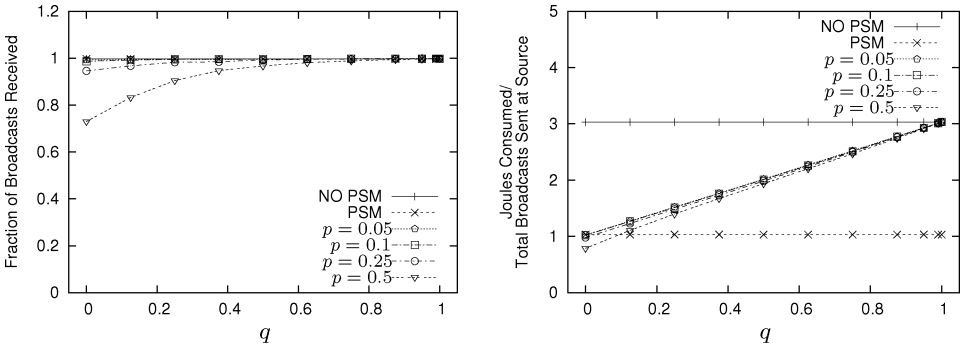
4.4 PBBF Performance in Random Networks with Nonideal MAC

The goal of our simulation study was to measure our success in meeting the design goals of PBBF and investigate the trade-off between energy, latency, and reliability in a more realistic setting. Essentially, we did the simulations to verify that the trends from Section 4.1 hold in random networks when collisions and interference are present. We implemented PBBF/IEEE 802.11 using the *ns-2* network simulator (ns-2—Network Simulator; go online to `http://www.isi.edu/nsnam/ns`).

Our implementation does not handle synchronization of nodes. Because IEEE 802.11 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 currently exist [Elson and Römer 2002], we assume perfect synchronization in the network. This is an assumption that other MAC protocols for sensors have made as well (e.g., Rajendran et al. [2003]). The

Table II. Simulation Parameter Values

Parameter	Value
N	50
q	0.25
Δ	10.0
Total packet size	64 bytes
Data packet payload	30 bytes



(a) Average number of broadcasts received as q varies.

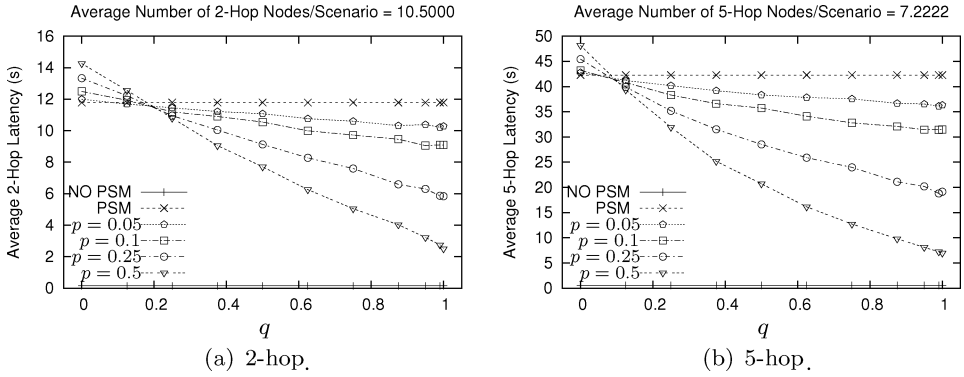
(b) Average energy consumption as q varies. The degree of overlap of PBBF lines increases with q.

Fig. 15. Average energy consumption and reliability as q varies for NO-PSM, PSM = IEEE 802.11 PSM, and PBBF/IEEE 802.11 PSM ($p = *$ lines).

length of the beacon interval, BI, and ATIM window, AW, are set according to the values of T_{frame} and T_{active}, respectively, in Table I. The bit rate of the nodes is 19.2 kb/s. In this section, the energy for transmissions, receptions and idling are accounted for and P_{TX}, P_I, P_S are the power levels of the sensor radio to transmit, receive/idle, and sleep, respectively (see Table I).

We evaluate the performance of PBBF with a simple broadcast application. For each scenario, one random node is chosen to be the broadcast source. A new broadcast is generated and sent deterministically at the source at a rate of λ broadcasts/s (see Table I). The total size and data payload of each packet are the same for all packets. Node density is $\Delta = \frac{\pi R^2 N}{A}$, where R is the range of a node, N is the number of nodes, and A is the area of the region where nodes are located (see Table II). To test PBBF in this setting, we varied the p and q values. We also tested PBBF when q was kept constant and Δ varied. We do not present these results here since they exhibit similar trends with the results in Section 4.4.1. We ran each simulation for 500 and each data point is averaged over 10 runs.

4.4.1 *The Impact of the q Parameter.* Figure 15(a) illustrates how the q value affects the fraction of broadcasts 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. Figure 15(b) shows how the average energy consumed at a node, normalized for the number of broadcasts generated, changes with q. We can see



(a) 2-hop. (b) 5-hop.

Fig. 16. (a) Two-hop and b) Five-hop average broadcast latency as q varies for NO-PSM, PSM = IEEE 802.11 PSM, and PBBF/IEEE 802.11 PSM ($p = *$ lines).

that using PSM saves almost 2 J/broadcast over using no PSM. The figure also shows that energy increases linearly with the q value. We also observe that q determines the energy usage because, regardless of the p value, the PBBF lines overlap.

Figures 16(a) and (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 source during the ATIM window, so they are sent with a delay of about AW. As expected, PSM consistently has a high latency ($\approx AW + BI$), 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. Essentially, as the reliability decreases, broadcasts are likely to traverse longer paths, and hence PBBF performs with higher 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 Figures 16(a) and (b), we can also see that the crossover q point where PBBF does better 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.

5. ADAPTIVE PBBF

The main goal of PBBF is to provide application designers tradeoff knobs, p and q, to achieve the desired operation points in terms of energy, latency, and reliability. In Section 4, assuming fixed values for p and q, we have shown the relationship between these knobs and the QoS of the broadcast (i.e., energy, latency, and reliability levels). While this study explains how to set the p and q parameters, it is also desirable to determine p and q with minimal support from the application designer. To this end, we propose *adaptive PBBF*, which adjusts p and q dynamically in response to feedback collected about the level of QoS achieved in the network. Adaptive PBBF is a heuristic-based protocol, which is composed of three components: (1) QoS specification, (2) feedback collection,

and (3) dynamic adaptation to build situation-awareness into PBBF. Through these components, adaptive PBBF gains the ability to perceive the network environment and modify its behavior to converge to the desired operation point. In the remainder of this section, we present these three components in more detail.

5.1 QoS Specification

To build QoS into any system involves a specification that captures application's requirements. The QoS parameters in adaptive PBBF are (1) energy used per broadcast (J), (2) latency per hop (s), and (3) reliability in terms of average percentage of nodes to receive the broadcasts. The application designer is required to specify two of these parameters, leaving the third free. For instance, constraints for latency and reliability may be defined, while letting PBBF minimize the energy consumption within these constraints. In the case when the QoS specification cannot be mapped into feasible p and q values, adaptive PBBF requires a priority order for the constraints such that the constraint with the higher priority is satisfied, while the second constraint is approximated as best as possible. If the requirements cannot be satisfied by any means, adaptive PBBF operates as a best-effort scheme.

5.2 Feedback Collection

To ensure the contracted QoS is sustained, it is essential to monitor QoS parameters and adjust accordingly in response to deviations. To this end, adaptive PBBF employs a feedback collection mechanism. Initially, the source announces the $p = p_{init}$ and $q = q_{init}$ with the first broadcast. (p_{init} and q_{init} can also be loaded to the sensors in the predeployment phase.) From this point on, the source node monitors the network performance by collecting feedback. However, to avoid feedback implosion, only a set of sensors, S, reports back to the source. The feedback comprises the average observed energy, latency, and reliability levels since the last time p and q changed. Specifically, a sensor $i \in S$ provides the following between two p and q updates: (1) the number of broadcasts received, B_i, (2) the energy used per broadcast, E_i/B_i, where E_i is the total energy consumption between two p and q updates, and, (3) average per-hop latency, L_i. To keep track of latency, each broadcast packet is timestamped with t_{send} by the source, and carries a hop count field, which is incremented at each hop. If a sensor i, is n hops away from the source, upon receiving jth broadcast at time t_{recv}, sensor i can calculate per-hop latency for broadcast j as $L_i(j) = (t_{recv} - t_{send})/n$. Sensor i reports L_i as $\sum_j L_i(j)/B_i$.

Based on sensor feedback, the source calculates reliability, energy, and latency achieved by the current p and q as $R_{feedback}$, $E_{feedback}$, and $L_{feedback}$ as follows:

$$R_{feedback} = \frac{\sum_i \frac{B_i}{B_{total}}}{|S|}, \quad E_{feedback} = \frac{\sum_i \frac{E_i}{B_i}}{|S|}, \quad \text{and} \quad L_{feedback} = \sum_i \frac{L_i}{|S|}. \tag{12}$$

where B_{total} is the number of broadcasts sent since the last time p and q changed and $|S|$ is the size of S.

To decide if it is necessary to update p and q, the source maintains an exponentially weighted moving average (EWMA) of the average energy, latency, and reliability reported during each feedback collection period. A feedback collection period continues for at least k broadcasts. Hence, if the QoS performance of the network during the last k broadcasts deviates from previous observations, p and q are updated to reflect this change. To determine which nodes should be in S, upon a broadcast reception all sensors periodically toss a coin to decide if they should report back to the source based on some probability. Assuming that each node receives at least one broadcast, such feedback collection can track reliability. In our performance evaluation, we evaluate sampling uncertainty and energy overhead of our feedback collection mechanism by comparing it to an oracle that provides information about all sensors in the network without any overhead.

5.3 Dynamic Adaptation

To assure agreed-upon QoS, adaptive PBBF adjusts p and q based on the current state of the network. The new p and q parameters, p_{new} and q_{new}, respectively, are announced with the next broadcast after feedback collection for at least k broadcasts. We use three algorithms depending on which is left as the free parameter:

—*Fixed-Energy-and-Latency (EL).* Reliability is the free parameter and p and q are determined based on $E_{feedback}$ and $L_{feedback}$.
—*Fixed-Energy-and-Reliability (ER).* Latency is the free parameter and p and q are determined based on $E_{feedback}$ and $R_{feedback}$.
—*Fixed-Latency-and-Reliability (LR).* Energy is the free parameter and p and q are determined based on $L_{feedback}$ and $R_{feedback}$.

Adaptive PBBF makes necessary adjustments based on the degree of QoS achieved as compared to QoS specification (i.e., E_{target}, L_{target}, and R_{target}). This can be quantified as follows:

$$E_{target} = (1+\alpha)\cdot E_{feedback}, \tag{13}$$

$$L_{target} = (1+\beta)\cdot L_{feedback}, \tag{14}$$

$$R_{target} = (1+\gamma)\cdot R_{feedback}. \tag{15}$$

If any adjustments are necessary, based on the free parameter, adaptive PBBF uses either EL, ER or LR to determine p_{new} and q_{new}. Next we present these algorithms in detail. The notation used in the algorithms is summarized in Table III.

5.3.1 *Algorithm Fixed-Energy-and-Latency (EL).* The goal of EL is to determine p_{new} and q_{new} to operate close to E_{target} and L_{target}. Setting p_{new} and q_{new} also determines the exact reliability level that can be provided by adaptive PBBF. The pseudocode of EL is shown in Figure 17.

Intuitively, if both α and β are approximately *zero* (using a hysteresis constant h), we can assume PBBF is able to provide the desired QoS. However, if $E_{feedback}$ is greater than E_{target}, PBBF needs to decrease q to decrease

Table III. Notation Used in the Algorithms

Parameter	Explanation
d_q	Step size for q
d_p	Step size for p
α	Deviation from E_{target}
β	Deviation from L_{target}
γ	Deviation from R_{target}
h	Hysteresis constant

```
                    UPDATE(parameter, x)
                 1  h > 0
                 2  if |x| > h
                 3     then if x > 0
EL(α, β)         4             then
 1  UPDATE(q, α) 5                  if parameter == q
 2  UPDATE(p, β) 6                     then d_q = d_q · (x − 1)
                 7                          Increase q
                 8                     else  d_p = d_p · (x − 1)
ER(α, γ)         9                          Decrease p
 1  UPDATE(q, α) 10             else
 2  UPDATE(p, γ) 11                  if parameter == q
                 12                     then d_q = d_q · (x − 1)
                 13                          Decrease q
                 14                     else  d_p = d_p · (x − 1)
                 15                          Increase p
```

Fig. 17. Algorithms Fixed-Energy-and-Latency (EL) and Fixed-Energy-and-Reliability (ER). In EL, reliability is the free parameter and p and q are determined based on $E_{feedback}$ and $L_{feedback}$. In ER, latency is the free parameter and p and q are determined based on $E_{feedback}$ and $R_{feedback}$.

energy consumption. On the other hand, if $E_{feedback}$ is less than E_{target}, PBBF increases q to increase reliability, which is the free parameter. Given q, q_{new} is calculated as

$$q_{new} = q + g \cdot d_q, \tag{16}$$

where d_q is the step size for q, and g is the direction of the update (i.e., $g = 1$ for increase and $g = -1$ for decrease). The step size d_q is initially set to 0.1. The step size continues to be updated based on the magnitude of difference between feedback and target parameters (see Figure 17).

PBBF adjusts p independently of q in a similar way. If $L_{feedback}$ is greater than L_{target}, PBBF needs to increase p to decrease latency. On the other hand, if $L_{feedback}$ is smaller than L_{target}, PBBF decreases p to increase reliability. Given p, p_{new} is determined as follows:

$$p_{new} = p + g \cdot d_p, \tag{17}$$

where d_p is the step size for p. The step size d_p is initially set to 0.1, but is updated based on β (see Figure 17). It must be noted that, although q_{new} impacts

```
LR(β, γ)
1  UPDATE(p, β)
2  c = p·q / (1−p)
3  UPDATE(p, γ)
4  q = c · (1−p) / p
```

Fig. 18. Algorithm Fixed-Latency-and-Reliability (LR). In LR, energy is the free parameter and p and q are determined based on $L_{feedback}$ and $R_{feedback}$. (The UPDATE($parameter, x$) algorithm is shown in Figure 17.)

both energy and latency (see Equations (7) and (8)), this impact is ignored. Essentially, counting for such an impact requires estimating L_1 (i.e., the channel access time) and L_2 (i.e., the time it takes to wake up neighbors upon reception of a broadcast) in Equation (8), which poses a significant challenge. However, once the algorithm converges to the desired energy consumption, E_{target}, it also converges to L_{target}.

5.3.2 *Algorithm Fixed-Energy-and-Reliability (ER).* The goal of ER is to determine p_{new} and q_{new} to operate close to E_{target} and R_{target}, which also determines the exact latency level. The pseudocode of ER is shown in Figure 17.

If both α and γ are approximately *zero* (using a hysteresis constant h), we can assume PBBF is able to provide the desired QoS. Otherwise, the value of q_{new} to achieve the desired E_{target} is calculated in the same way as EL, and therefore is not repeated here. PBBF adjusts p based on $R_{feedback}$ and R_{target}. If $R_{feedback}$ is less than R_{target}, PBBF needs to decrease p to increase the reliability level. On the other hand, if $R_{feedback}$ is greater than R_{target}, PBBF increases p to decrease latency. We calculate p_{new} to achieve the desired R_{target} the same way as in EL.

As in EL, reliability, similar to latency, is also affected by q_{new}. However, counting for this effect requires knowledge of critical bond probability (see Remark 4.1). However, to the best of our knowledge, no published results exist for critical bond probability in random networks in percolation theory. Therefore, ignoring this effect, ER independently sets p_{new} and q_{new}.

5.3.3 *Algorithm Fixed-Latency-and-Reliability (LR).* The goal of LR is to determine p_{new} and q_{new} to operate close to L_{target} and R_{target}. The pseudocode of LR is shown in Figure 18.

If both β and γ are approximately *zero* (using a hysteresis constant h), we can assume PBBF is able to provide the desired QoS. If $L_{feedback}$ is greater than L_{target}, PBBF needs to increase p to decrease latency. However, if $L_{feedback}$ is less than L_{target}, PBBF decreases p, which would increase the reliability level. We calculate p_{new} to achieve L_{target} in the same way as in EL. However, after setting $p = p_{new}$ to reflect the desired latency, LR continues tuning p and q parameters until R_{target} is achieved. Essentially, if $R_{feedback}$ is less than R_{target}, PBBF needs to decrease p and increase q while keeping $\frac{p \cdot q}{1-p}$ constant to increase the reliability level without affecting latency. Essentially, we derive

the relationship between p and q to keep latency constant based on Equation (8)

$$L = L_1 + L_2 \cdot \frac{1-p}{1-p+p\cdot q} = L_1 + L_2 \cdot \frac{1}{1+\frac{p\cdot q}{1-p}}. \quad (18)$$

Furthermore, if $R_{feedback}$ is greater than R_{target}, PBBF increases p and decreases q while keeping $\frac{p\cdot q}{1-p}$ constant to improve energy consumption. The calculation of p_{new} and q_{new} for the corresponding reliability level is similar to ER, and therefore is not repeated here.

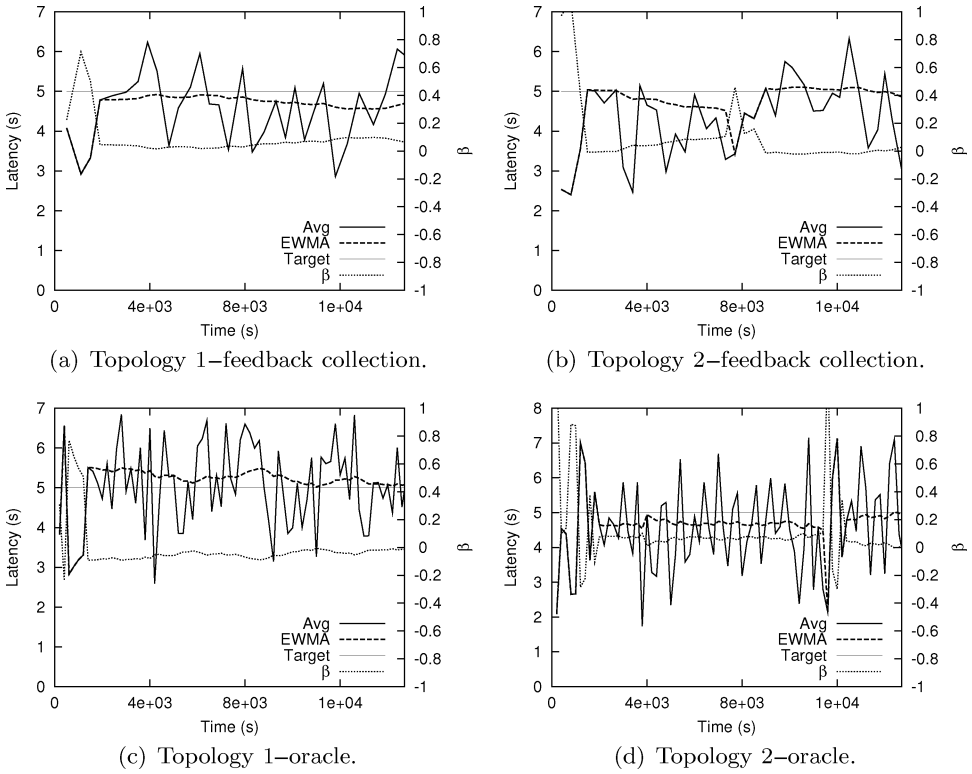
6. EVALUATION OF ADAPTIVE PBBF

The goal of our evaluation is to show that adaptive PBBF can dynamically adjust p and q to sustain the QoS specification. To this end, we study the performance of adaptive PBBF/IEEE 802.11 PSM via simulations. Additionally, we compare the performance of feedback collection against results using an oracle. Our experiments are similar in setting to the experiments presented in Section 4.4. The parameters specific to adaptive PBBF are as follows. To adapt p and q, the source collects feedback until at least 20 reports are received. Each sensor node sends a report with probability 0.2. After a feedback collection period, the source announces the p_{new} and q_{new} with the next broadcast. Since the feedback obtained after each feedback collection period might have high variance, the source maintains an EWMA of energy, latency, and reliability. The initial p and q, p_{init}, and q_{init}, respectively, are set to 0.5. QoS specification is given as $E_{target} = 2$ J, $L_{target} = 5$ s, and $R_{target} = 0.85$. Each simulation runs for 15,000 s. We present results for two different topologies, Topology 1 and Topology 2, to illustrate the convergence of the algorithms in different networks.

6.1 Performance of Fixed-Energy-and-Latency (EL)

The goal of EL is to adapt p and q to operate as close to $E_{target} = 2$ J and $L_{target} = 5$ s as possible. Figures 19(a) and (b) illustrate the average and EWMA latency after each feedback collection period. We observe that, in the presence of sampling uncertainty, the network is still able to sustain L_{target}. On the other hand, using perfect information from the oracle, adaptive PBBF reacts more drastically to changes in the network, whereas feedback collection seems to have a smoothing effect on adaptivity behavior. However, using an oracle allows higher convergence speeds compared to feedback collection case (see Avg and β in Figures 19(c) and (d)).

EL is more successful in maintaining E_{target} (see Figure 20) compared to the latency performance. This can be also observed from the convergence of p and q for the no oracle case (see Figure 21). While $q = 0.5$ seems to be the right value to achieve E_{target}, EL eventually decreases p to increase reliability while maintaining latency close to L_{target}. This can be clearly seen for Topology 2 in Figure 19(b) at around 1000 s and 7000 s. At both instances p is reduced to improve latency. Furthermore, the energy consumption due to feedback collection is negligible compared to oracle simulations. The average energy consumption per broadcast is successfully maintained at 2 J in both cases (i.e., feedback collection and oracle; see Figure 20).



(a) Topology 1–feedback collection.

(b) Topology 2–feedback collection.

(c) Topology 1–oracle.

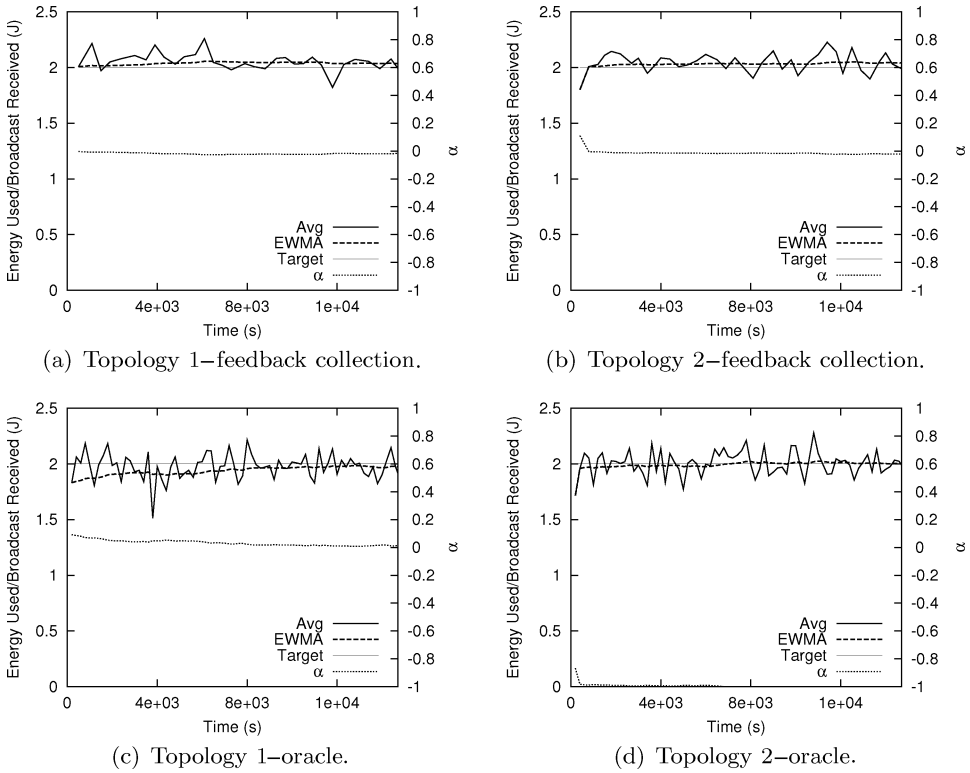
(d) Topology 2–oracle.

Fig. 19. Fixed-Energy-and-Latency (EL): latency under different topologies using feedback collection ((a) and (b)) and using an oracle ((c) and (d)).

In addition to satisfying QoS constraints, EL achieves 100% reliability most of the time with feedback collection (see Figure 22). The oracle simulations show similar behavior in terms of reliability, and therefore are omitted. Since we observe similar performance trends in the comparison of feedback collection versus oracle for both ER and LR, we do not present any oracle results in the rest of this section.

6.2 Performance of Fixed-Energy-and-Reliability (ER)

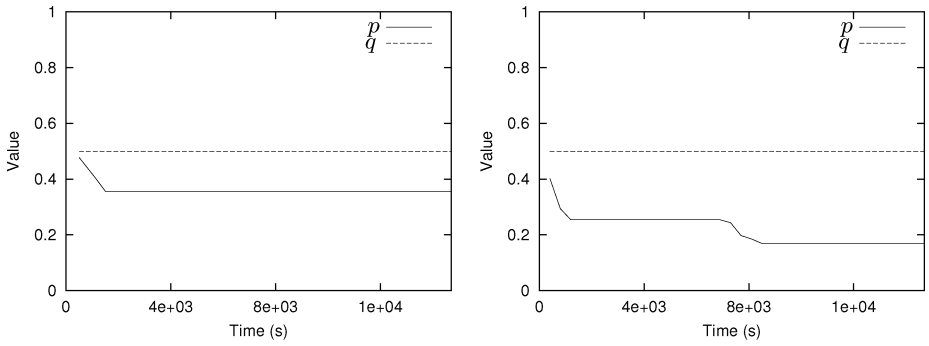
The goal of ER is to adapt p and q to operate as close to $E_{target} = 2$ J and $R_{target} = 0.85$ as possible. ER is able to achieve E_{target} and R_{target} with $p_{init} = 0.5$ and $q_{init} = 0.5$ (see Figures 23 and 24). Hence, in both simulations with different topologies, q value is not updated throughout the simulation runs. However, in Topology 2, adaptive PBBF chooses to increase p to improve latency as long as the reliability is maintained higher than R_{target} (see Figures 25(b) and 26(b)). Essentially, since the network achieves a higher reliability than R_{target} ($\gamma \approx -0.2$ until 4000 s; see Figure 24(b)), this provides substantial room for improving latency. We do not observe a similar reaction in Topology 1 since the average reliability is maintained close to $R_{target} = 0.85$ with current p and q values and there is no room for improving latency (see Figures 24(a) and 26(a)).



(a) Topology 1–feedback collection. (b) Topology 2–feedback collection.

(c) Topology 1–oracle. (d) Topology 2–oracle.

Fig. 20. Fixed-Energy-and-Latency (EL): energy consumption under different topologies using feedback collection ((a) and (b)) and using an oracle ((c) and (d)).

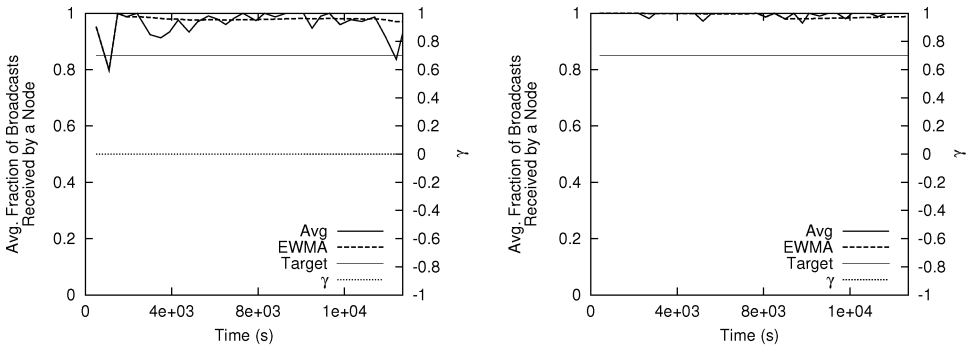


(a) Topology 1. (b) Topology 2.

Fig. 21. Fixed-Energy-and-Latency (EL): the convergence of p and q values.

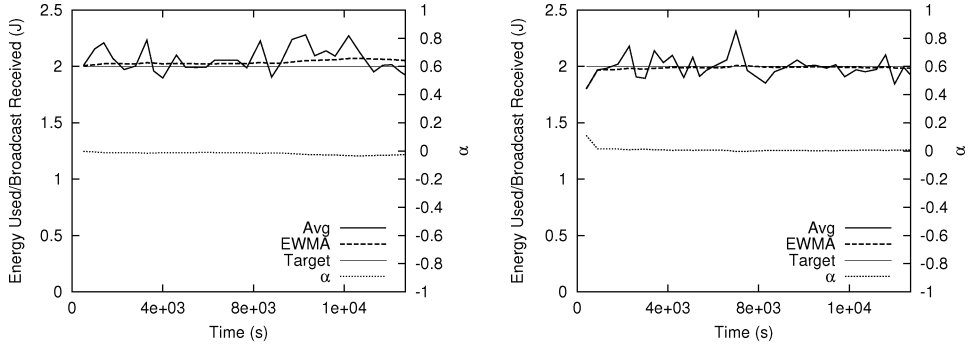
6.3 Performance of Fixed-Latency-and-Reliability (LR)

The goal of LR is to adapt p and q to operate as close to $L_{target} = 5$ s and $R_{target} = 0.85$ as possible. LR is different than EL and ER in the sense that p and q are not independent from each other. Essentially, q is set based on the p value that keeps $\frac{p \cdot q}{1-p}$ constant. This is necessary since any change made in



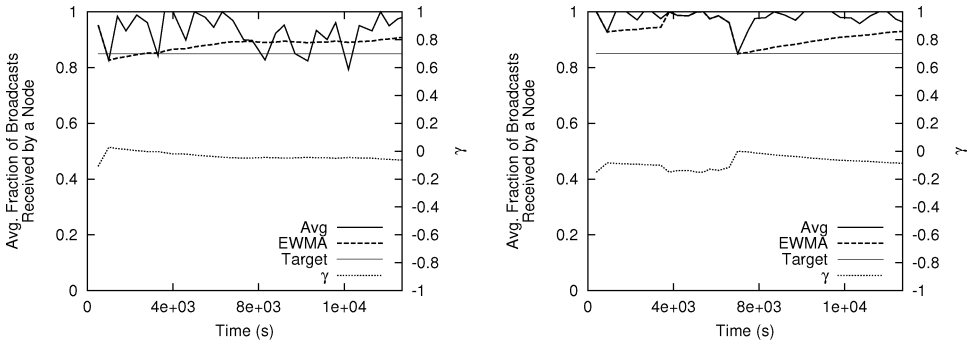
(a) Topology 1. (b) Topology 2.

Fig. 22. Fixed-Energy-and-Latency (EL): reliability.



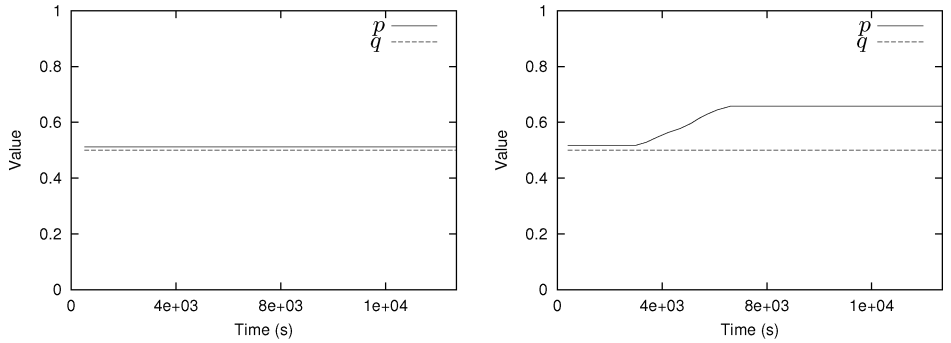
(a) Topology 1. (b) Topology 2.

Fig. 23. Fixed-Energy-and-Reliability (ER): energy consumption.



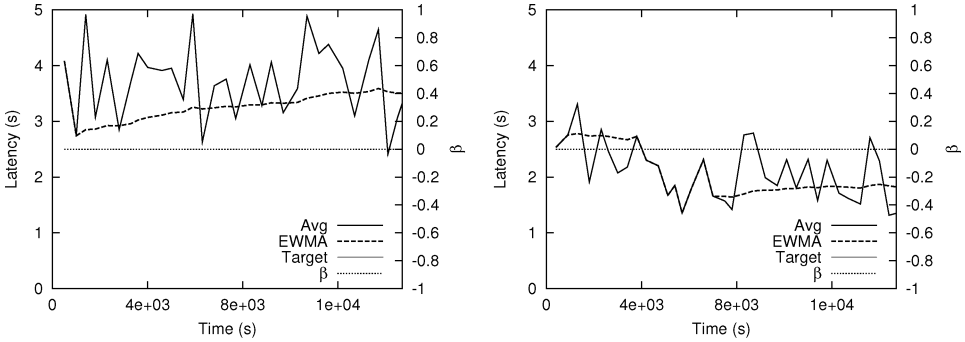
(a) Topology 1. (b) Topology 2.

Fig. 24. Fixed-Energy-and-Reliability (ER): reliability.



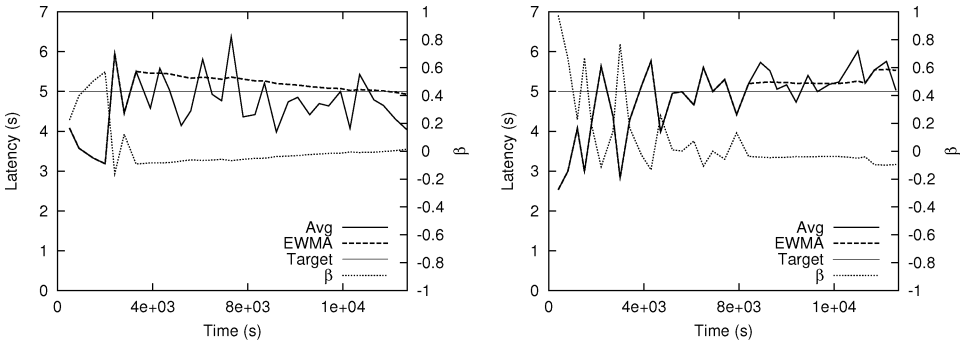
(a) Topology 1. (b) Topology 2.

Fig. 25. Fixed-Energy-and-Reliability (ER): the convergence of p and q values.



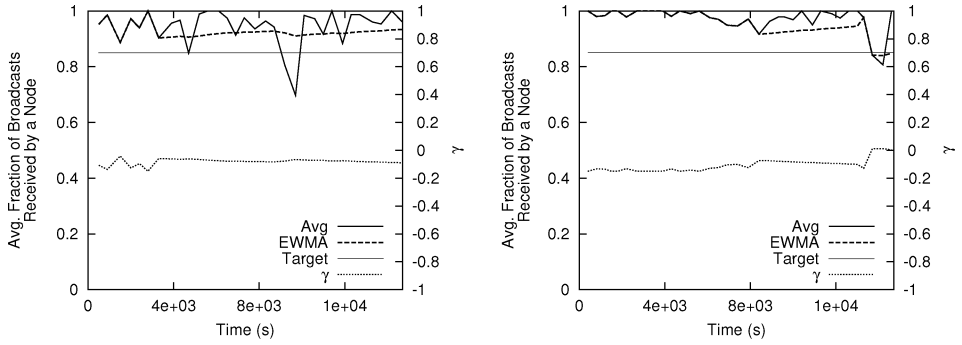
(a) Topology 1. (b) Topology 2.

Fig. 26. Fixed-Energy-and-Reliability (ER): latency.



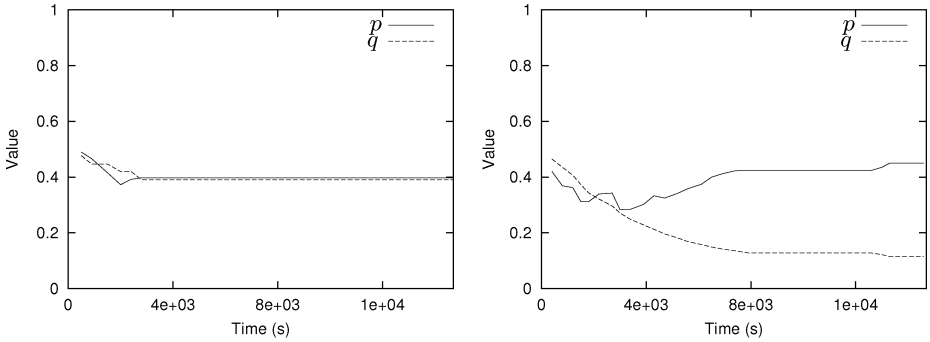
(a) Topology 1. (b) Topology 2.

Fig. 27. Fixed-Latency-and-Reliability (LR): latency.



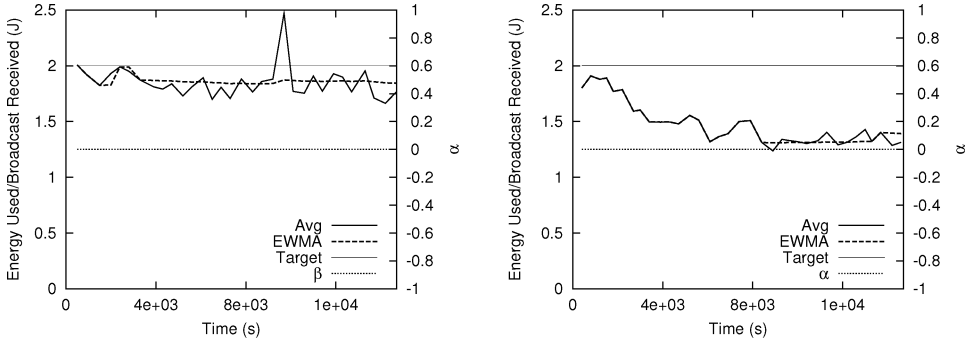
(a) Topology 1. (b) Topology 2.

Fig. 28. Fixed-Latency-and-Reliability (LR): reliability.



(a) Topology 1. (b) Topology 2.

Fig. 29. Fixed-Latency-and-Reliability (LR): the convergence of p and q values.



(a) Topology 1. (b) Topology 2.

Fig. 30. Fixed-Latency-and-Reliability (LR): energy consumption.

p and q affects both latency and reliability, while q additionally determines energy consumption.

Simulation results show that for Topology 1, latency and reliability converge to desired values between 3000–4000 s (see Figures 27(a) and 28(a)). Once this operation point is achieved, p and q are not updated and the network maintains a consistent energy consumption history (see Figures 29(a) and 30(a)). For Topology 2, p eventually stays the same while q constantly decreases until both parameters converge around 8000 s (see Figure 29(b)). Essentially, in Topology 2, adaptive PBBF finds a chance to improve energy consumption while keeping latency and reliability close to the target values (i.e., $\beta > 0$ and $\gamma < 0$ and approximately 0 to sustain desired QoS) (see Figures 27(b), 28(b), and 30(b)). However, in Topology 1, once p and q converge such that L_{target} and R_{target} are satisfied, adaptive PBBF stops modifying p and q to improve energy consumption.

7. CONCLUSION

We have presented and evaluated through analysis and simulations the performance of a probabilistic broadcast protocol (PBBF) for multihop WSNs. We have quantified the energy-latency tradeoff at a given level of reliability using PBBF. This is attained by allowing an application designer to tune the values of parameters p and q while maintaining the value of $1 - p \cdot (1 - q)$ above the threshold required to achieve very high reliability. We have implemented the PBBF protocol in *ns-2* and studied its performance characteristics for a generic broadcast application. Our experiments indicate that PBBF is an efficient broadcast mechanism. PBBF provides an application designer the opportunity to tune the system to an appropriate operating point along the reliability-resource-performance spectrum. Furthermore, we proposed an extension to PBBF, adaptive PBBF, which dynamically adjusts p and q based on QoS specification determining any two of energy, latency, and reliability parameters. Our simulation study shows that adaptive PBBF successfully converges to an operating point that satisfies the application requirements reasonably fast and continues to improve performance based on the free parameter.

REFERENCES

ELSON, J. AND RÖMER, K. 2002. Wireless sensor networks: A new regime for time synchronization. In *Proceedings of ACM Hot Topics in Networks 2002*.

GRIMMET, G. AND STACEY, A. M. 1998. Critical probabilities for site and bond percolation models. *Ann. Prob. 26*, 4, 1788–1812.

GUTTMANN, A. J. AND BURSILL, R. J. 1990. Critical exponent for the loop erased self-avoiding walks by monte carlo methods. *J. Statist. Phys. 59*.

HAAS, Z. J., HALPERN, J. Y., AND LI, L. 2002. Gossip-based ad hoc routing. In *Proceedings of IEEE Infocom*.

HEINZELMAN, W. R., KULIK, J., AND BALAKRISHNAN, H. 1999. Adaptive protocols for information in wireless sensor networks. In *Proceedings of ACM MobiCom 1999*.

IEEE 802.11. 1999. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE, Piscataway, NJ.

KENYON, R. 1999. Loop-erased random walks, Universite de Paris-Sud, Paris, France. (Talk given on May 31.)

LEVIS, P., PATEL, N., CULLER, D., AND SCHENKER, S. 2004. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st Symposium on Networked System Design and Implementation* (NSDI).

LI, Y., YE, W., AND HEIDEMANN, J. 2005. Energy and latency control in low duty cycle MAC protocols. In *Proceedings of the IEEE Wireless Communications of Networking Conference* (WCNC).

LU, C., XI, G., CHIPARA, O., FOK, C.-L., AND BHATTACHARYA, S. 2005. A spatiotemporal query service for mobile users in sensor networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems* (ICDCS).

NEWMAN, M. E. J. AND ZIFF, R. M. 2001. A fast Monte Carlo algorithm for site or bond percolation. Tech. rep. Santa Fe Institute., Santa Fe, NM.

POLASTRE, J., HILL, J., AND CULLER, D. 2004. Versatile low power media access for wireless sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems* (SenSys).

RAJENDRAN, V., OBRACZKA, K., AND GARCIA-LUNA-ACEVES, J. J. 2003. Energy-efficient, collision-free medium access control for wireless sensor networks. In *Proceedings of the ACM SenSys 2003*.

REIJERS, N. AND LANGENDOEN, K. 2003. Efficient code distribution in wireless sensor networks. In *Proceedings of the ACM WSNA 2003*.

SASSON, Y., CARIN, D., AND SCHIPER, A. 2003. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of the IEEE Wireless Communications of Networking Conference* (WCNC).

SCHURGERS, C., TSIATSIS, V., GANERIWAL, S., AND SRIVASTAVA, M. 2002. Topology management for sensor networks: exploiting latency and density. In *Proceedings of the ACM MobiHoc 2002*.

SINHA, P., SIVAKUMAR, R., AND BHARGHAVAN, V. 2001. Enhancing ad hoc routing with dynamic virtual infrastructures. In *Proceedings of the IEEE Infocom*.

SIVAKUMAR, R., SINHA, P., AND BHARGHAVAN, V. 1999. CEDAR: A core-extraction distributed ad hoc routing algorithm. In *Proceedings of the IEEE Infocom*.

STATHOPOULOS, T., HEIDEMANN, J., AND ESTRIN, D. 2003. A remote code update mechanism for wireless sensor networks. Tech. Rep. CENS-TR-30. Center for Embedded Networked Computing., UCLA, Los angeles, CA.

VAN DAM, T. AND LANGENDOEN, K. 2003. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the ACM SenSys 2003*.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*.

YE, W., SILVA, F., AND HEIDEMANN, J. 2006. Ultra-low duty cycle MAC with scheduled channel polling. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems* (SenSys).

Received July 2005; revised September 2006, April 2007, October 2007; accepted December 2007