On Reliable Broadcast in Low Duty-Cycle Wireless Sensor Networks

Feng Wang, *Student Member*, *IEEE*, and Jiangchuan Liu, *Senior Member*, *IEEE*

Abstract—Broadcast is one of the most fundamental services in wireless sensor networks (WSNs). It facilitates sensor nodes to propagate messages across the whole network, serving a wide range of higher level operations and thus being critical to the overall network design. A distinct feature of WSNs is that many nodes alternate between active and dormant states, so as to conserve energy and extend the network lifetime. Unfortunately, the impact of such cycles has been largely ignored in existing broadcast implementations that adopt the common assumption of all nodes being active all over the time. In this paper, we revisit the broadcast problem with active/dormant cycles. We show strong evidence that conventional broadcast approaches will suffer from severe performance degradation, and, under low duty cycles, they could easily fail to cover the whole network in an acceptable time frame. To this end, we remodel the broadcast problem in this new context, seeking a balance between efficiency and latency with coverage guarantees. We demonstrate that this problem can be translated into a graph equivalence, and develop a centralized optimal solution. It provides a valuable benchmark for assessing diverse duty-cycle-aware broadcast strategies. We then extend it to an efficient and scalable distributed implementation, which relies on local information and operations only, with built-in loss compensation mechanisms. The performance of our solution is evaluated under diverse network configurations. The results suggest that our distributed solution is close to the lower bounds of both time and forwarding costs, and it well resists to the wireless loss with good scalability on the network size and density. In addition, it enables flexible control toward the quality of broadcast coverage.

Index Terms—Broadcast, reliability, duty cycle, wireless sensor networks, time-coverage graph.

1 Introduction

Broadcast is one of the most fundamental services in wireless sensor networks (WSNs) [3]. It facilitates sensor nodes to propagate messages across the whole network, serving a wide range of higher level operations: during networking configuration, control messages may be broadcast from the sink to all sensor nodes; for data collection, interest or query messages may be broadcast within the network; upon observing an event, a sensor node may broadcast a message to coordinate with other nodes for tracing the event and storing sensed data; to name but a few. Hence, implementing an effective network-wide broadcast service is critical to the overall performance optimization of a WSN.

Flooding and gossiping [3] are two commonly used broadcast approaches, though their basic forms are known inefficient. If we assume all network nodes are active during the broadcast process (referred to as *all-node-active assumption*), ideally every node needs to receive and forward the broadcast message at most once. Significant efforts, thus, have been made toward enhancing the efficiency of the basic flooding or gossiping, while retaining their robustness in the presence of error-prone transmissions [11], [20].

The all-node-active assumption is valid for wired networks and for many conventional multihop wireless networks. It, however, fails to capture the uniqueness of energy-constrained wireless sensor networks. The sensor nodes are often alternating between *dormant* and *active* states [6], [14], [24], [25]; in the former, they go to sleep and thus consume little energy, while in the latter, they actively perform sensing tasks and communications, consuming significantly more energy (e.g., 56 mW for IEEE802.15.4 radio plus 6 to 15 mW for Atmel ATmega 128L microcontroller and possible sensing devices on a MicaZ mote). Define *duty cycle* as the ratio between active period and the full active/dormant period. A low duty-cycle WSN clearly has a much longer lifetime for operation, but breaks the all-node-active assumption. In such a network, if the number of nodes is very small, it may be possible to wake up all nodes for broadcast through global synchronization with customized active/dormant schedules. For larger scale WSNs, however, synchronization itself remains an open problem. More importantly, the duty cycles are often optimized for the given application or deployment, and a broadcast service accommodating the schedules is, thus, expected for cross-layer optimization of the overall system.

In this paper, we revisit the broadcast problem in low duty-cycle WSNs. Their scale, together with their application/deployment-specific duty cycles, renders the all-node-active assumption impractical. This in turn introduces a series of new challenges toward implementing network-wide broadcast. From a local viewpoint, since the neighbors of a node are not active simultaneously, a node would have to forward a message multiple times at different instances; from a global viewpoint, since the topology is time varying with no persistent connectivity, if not well planned, the latency for a message to reach all nodes can be significantly prolonged. The error-prone wireless links further aggravate these problems. Our experiments (Section 6.2) have shown

• *The authors are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada. E-mail: {fwa1, jcliu}@cs.sfu.ca.*

Manuscript received 28 June 2010; revised 30 Jan. 2011; accepted 18 Mar. 2011; published online 28 Apr. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-06-0318. Digital Object Identifier no. 10.1109/TMC.2011.94.

that, for ultra low duty cycles, a conventional broadcast strategy would simply fail to cover all the nodes within an acceptable time frame.

To this end, we remodel the broadcast problem in this new context, seeking a balance between efficiency and delay with reliability guarantees. We demonstrate that this problem can be translated into a graph equivalence, and develop a centralized optimal solution. It provides a valuable benchmark for assessing diverse duty-cycle-aware broadcast strategies. We then extend it to an efficient distributed implementation, which relies only on local information and operations, with inherent loss compensation mechanisms.

We evaluate our solution under diverse network configurations. The results suggest that our distributed solution is close to the practical lower bounds of both time and forwarding costs, and it well resists to the wireless loss with good scalability on the network size and density. In addition, it enables flexible control toward the quality of broadcast coverage.

The remainder of this paper is organized as follows: Section 2 lists the related work. In Section 3, we reformulate the broadcast problem in low duty-cycle WSNs. We introduce a centralized optimal solution in Section 4. It is then extended to a scalable and robust distributed implementation in Section 5. In Section 6, we present extensive simulation results to evaluate the performance of our solution. We further discuss some key practical issues in Section 7, and concludes the paper in Section 8.

2 Background and Related Work

There have been numerous studies on broadcast in wired networks and in wireless ad hoc networks [4], [5], [16]. While the commonly used flooding and gossiping remain basic approaches for wireless sensor networks, substantial revisions are needed to accommodate the challenges from this new network environment [3]. An example is Smart Gossip [11], which extends the basic gossip to minimize forwarding overhead. To determine the forwarding probability for each sensor node, the algorithm keeps tracking previous broadcasts and adaptively adjusts the probability to match the topological properties among the sensor nodes. In [9], a timing heuristic named Forwarding-node Declaration Latency (FDL) is proposed to reduce redundant message forwardings in the basic flooding. The idea is to let a node defer a forwarding with a latency proportional to its residual energy. A more recent work is Robust Broadcast Propagation (RBP) [20], which extends the flooding-based approach and targets for reliable broadcast. It lets each node flood the received broadcast message only once, and then by overhearing and explicit ACKs, the node may perform retransmissions for local repairs. Both the retransmission thresholds and the number of retries depend on the node density and topology information gathered from previous rounds of broadcast.

There are other related works on code redistribution and update propagation in WSNs, such as Trickle [12]. Their emphasis is on the distribution of the newest version of the code; the update frequency is much lower than that of a generic broadcast service working for many higher level operations.

Our work, different from the aforementioned, considers a more realistic scenario with sensor nodes alternating between active and dormant states to save energy. In this scenario, previous works may fail or suffer from poor performance due to the invalidation of the *all-node-active* assumption.

There have been recent works investigating low duty-cycle wireless sensor networks [15], [7], [22], [8], [21]. Among them, Probability-Based Broadcast Forwarding (PBBF) [15] implements a MAC layer solution for flooding in low duty-cycle sensor networks and investigates tradeoffs among flooding reliability, latency, and energy consumption. To handle dynamic traffic loads, Sun et al. [22] present Receiver-Initiated MAC (RI-MAC), which strives to minimize the time that a sender and its intended receiver occupy the wireless medium and to find a rendezvous time for exchanging data, while still decoupling the sender and receiver's duty-cycle schedules. Later, they further propose Asynchronous Duty-cycle Broadcasting (ADB) [21] to enhance asynchronous duty-cycling MAC protocols such as RI-MAC, so as to achieve efficient broadcast over multihop wireless sensor networks. These works have focused on the MAC layer, where the operations are of much shorter time scales and the duty cycles are subject to change with network traffic.[1] Our work, however, assumes that the active/dormant schedules are optimized and predetermined by the given application or deployment and should be strictly followed by the sensor nodes. In this context, Gu and He propose Dynamic Switch-based Forwarding (DSF) [7] to consider data forwarding in low duty cycles, which only addresses unicast from a data source to a sink. Guo et al. [8] further propose using unicasts to implement flooding in extremely low duty-cycle wireless sensor networks, where broadcast messages are forwarded by unicasts along energy optimal trees, and early opportunistic transmissions are conducted outside of trees to reduce the broadcast delays. Our work complements them by considering the scenario of providing a network-wide broadcast service. Our solution captures the unique features of the active/dormant schedules at sensor nodes and also makes effective use of local broadcast of wireless medium, achieving a balance between efficiency and latency with coverage guarantees.

3 Problem Statement

In this section, we reformulate the broadcast problem in low duty-cycle wireless sensor networks. To reflect the operation nature of real sensor products [1], [2] and also to simplify exposition, we divide time into equal-length *slots*.[2] The active and dormant periods are both integer multiples of time slots, and in each slot, an active node can either receive or forward one message only.

We do not assume any specific active/dormant schedule in our model. This brings two advantages: first, our solution

1. For example, a receiver may temporarily keep its radio on for a possible incoming message, or a sender may keep its radio on and wait for the receiver's radio to wake up.

2. We do not confine the length of one time slot into a specific value as long as it is acceptable for active/dormant state switching and message forwarding. As such, it can be tens to hundreds of milliseconds depending on the application and hardware configurations.

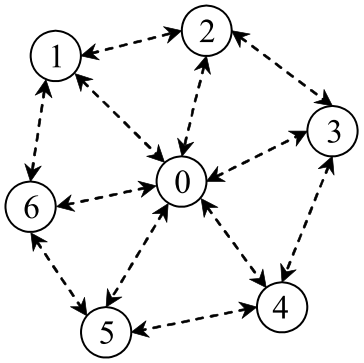

Fig. 1. An illustrative example for duty-cycle-aware broadcast. We use dashed lines for communications links, reflecting that they are not always available in the presence of duty cycles.

is generally applicable to diverse schedules; and second, our solution provides a generic tool for cross-layer optimization, i.e., for the collaborative optimization between active/dormant schedule and broadcast service.

3.1 Motivation

We first consider a motivational toy example shown in Fig. 1, where sensor node 0 needs to broadcast a message to all other nodes (1 through 6). Assume that there is no wireless loss,[3] it is easy to find a simple schedule: node 0 waits until all neighbors wake up and then forward the message. This strategy has the minimum message cost, i.e., only one message is forwarded. However, since the nodes' active/dormant patterns may be noticeably different from each other, it would take a very long time for all of them becoming active. In the worst, if there is no overlap among their active periods, the time become infinity, i.e., the strategy does not work.

An alternative is that node 0 forward the message as soon as one neighbor wakes up. The latency to accomplish broadcast is, thus, bounded by the time that the last neighbor turns active, together with node 0. Node 0, however, has to forward the same message six times in the worst case.

The problem is further complicated if the broadcast is more than one hop; for example, if node 1 needs to broadcast a message to all others. In this case, for node 4 to receive the message, the shortest path is through node 0, i.e., a 2-hop path, if all nodes are active. In low duty-cycle networks, however, node 0 may wake up very late, and in turn that a 3-hop path through nodes 2 and 3 or that through 6 and 5 might be faster. When the network size increases, the difference can be more remarkable, and, with higher chances, an all-node-active-based solution will fail to cover the whole network.

3.2 Problem Formulation

We now give a formal description of the duty-cycle-aware broadcast problem in wireless sensor networks. We will focus on the broadcast of a single message with a unique identifier (ID) from one source to all other nodes. By assigning different identifiers, our solution can be easily extended to broadcast a series of messages or broadcast messages from multiple sources. We assume there are n nodes in the network, indexed from 1 to n. For node i, $X_i(t)$ denotes its active/dormant state at time t, where $X_i(t) = 1$ if it is active and $X_i(t) = 0$ if it is dormant.

We represent the set of 1-hop neighbors of node i by N_i, i.e., those that can be directly covered by a message forwarding from node i if they are active. Here, we call 1-hop message broadcast from a node to its neighbors as "*forward*," so as to distinguish from our interest of network-wide broadcast (or *broadcast* in short).

Without loss of generality, we assume that the message is to be broadcast from node s, starting from time t_0. Let (u_i, t_i) denote the ith forwarding, where node u_i forward the message at time t_i, and C_i be the set of nodes that receive the broadcast message in the ith forwarding, the problem can be formulated as follows:

The Duty-Cycle-Aware Broadcast Problem. Given node s to broadcast a message starting from time t_0, find a forwarding schedule

$$S = \{(u_1, t_1), \ldots, (u_m, t_m)\} \quad (t_0 \le t_1 \le \cdots \le t_m)$$

that minimizes $f(|S|, t_m - t_0)$, a function of the total message forwarding cost ($|S|$) and the total latency ($t_m - t_0$).

The sequence should satisfy the following constraints:

1. *Duty-cycle constraint*:

$$X_{u_i}(t_i) = 1,$$
$$C_0 = \{s\}, \quad C_i = \{j | j \in N_{u_i}, \quad X_j(t_i) = 1\};$$

2. *Forwarding order constraint*:

$$u_1 = s,$$
$$\exists j, \quad t_j < t_i, \quad u_i \in C_j, \quad i = 2, 3, \ldots, m;$$

3. *Reliability constraint*:

$$\left| \bigcup_{i=0}^{m} C_i \right| = n.$$

The duty-cycle constraint follows that an active node u_i can successfully deliver the message to its neighbor, node j, at time t_i only if j is active at that time. The forwarding order constraint implies that the message is forwarded hop-by-hop, and only a node that has previously received the message can forward it. Finally, the reliability constraint ensures that all the nodes will be reached by the broadcast message. This last constraint can be relaxed to achieve flexible reliability requirements, as will be discussed in Section 7.

The objective function depends on the forwarding cost and the latency, and is in general specified by the target application. In this paper, we will focus on a common linear combination, $f(|S|, t_m - t_0) = \alpha|S| + \beta(t_m - t_0)$. By assigning different weights (α, β), it covers the demands from a broad spectrum of applications. For example, if the broadcast message is about an emergency event and of small size, a small α with a large β will ensure that the message is quickly delivered to the whole network, though possibly with higher forwarding costs. On the other hand, for large nonurgent messages, such as a code update, a large α with a

3. For ease of exposition, wireless communication losses are not considered at this stage. Loss-tolerant mechanisms will be presented in Section 5.

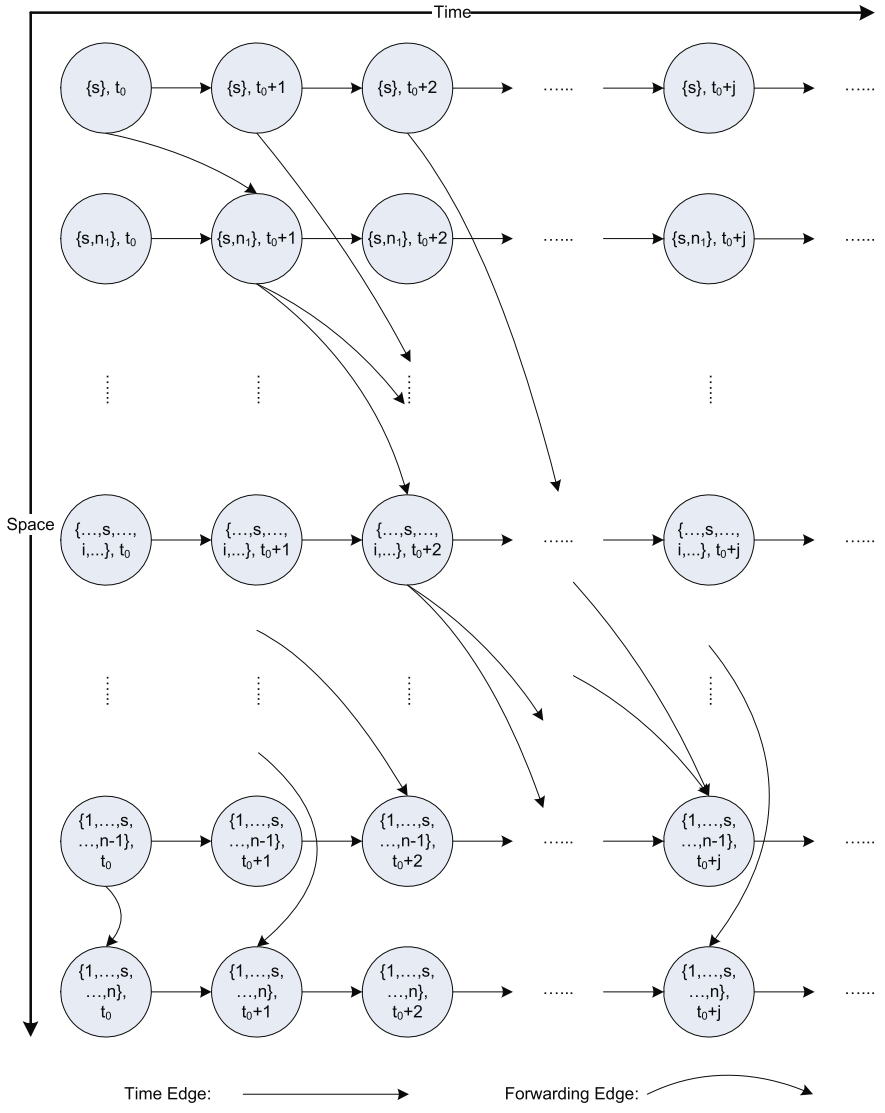

Fig. 2. An illustration of a constructed time-coverage graph.

small β will work well to save forwarding costs, and thus energy. It is worth noting that the optimal forwarding sequence, and hence its message and time costs, actually depends on the ratio α/β, while not their absolute values. We will examine the impact of this ratio and recommend practical settings in Section 6.1.

4 Centralized Optimal Solution

We first transform the duty-cycle-aware broadcast problem into a shortest path problem in a time-coverage graph. Assume that there is no wireless loss, given the network topology and the active/dormant patterns, this graph problem is solvable through a centralized dynamic programming algorithm. Its design principle also motivates the distributed implementation to be presented in the next section.

4.1 Problem Transformation: Shortest Path to Last Row

We construct a directed graph $G(V, E)$ as shown in Fig. 2, where its vertices are organized in two dimensions, indexed by time and space coverage, respectively. A vertex $v_{R,t}$ represents that, at time t, the sensor nodes in set R have received the broadcast message, i.e., being *covered*. The index R starts from $\{s\}$, and expands until it becomes $\{1, \ldots, n\}$. Obviously, each index R corresponds to a connected subnetwork in the original wireless sensor network and must include node s. Hence, although there are 2^n subsets of $\{1, \ldots, n\}$, the number of valid Rs are much less.

There are two kinds of edges in the graph, referred to as *time edges* and *forwarding edges*, respectively. A time edge connects two neighboring vertices along a row, from the earlier to the later. It corresponds to the case that no node in R will forward the message at a time t, and the same coverage state is, thus, inherited by the next time slot. A forwarding edge, on the other hand, corresponds to forwarding events. Specifically, a forwarding edge from $v_{R,t}$ to $v_{R',t'}$ means that, at time t, one or more active nodes in R will forward the message, which leads to a new coverage status R'. Clearly, we have $R \subset R'$, and $R' - R$ is the set of nodes that newly receive the message in this round of forwarding. We set $t' = t + 1$ as the time index for the destination vertex in the graph, which follows that a node

can only forward the newly received message in the next time slot or later. The only exception is for vertices in the last row, which corresponds to the full coverage with no further forwarding being necessary, and we, thus, set $t' = t$.

This time-coverage graph can be naturally related to the duty-cycle-aware broadcast problem: each forwarding sequence corresponds to a path from $v_{\{s\},t_0}$ to a vertex in the last row, and vice versa.

For objective function $f(S, t_m - t_0) = \alpha|S| + \beta(t_m - t_0)$, we assign weight β to each time edge since a delay of one time unit is incurred, and weight $(\alpha p + \beta(t' - t))$ to a forwarding edge from $v_{R,t}$ to $v_{R',t'}$, where p is the number of nodes in R that forward the message at time t. It is clear to see that the duty-cycle-aware broadcast problem is translated into the shortest path problem from $v_{\{s\},t_0}$ to a last-row vertex in the weighted graph.

4.2 Dynamic Programming Algorithm

Let $W(v_{R,t}, v_{R',t'})$ denote the weight of the edge from $v_{R,t}$ to $v_{R',t'}$, and $W(v_{R,t}, v_{R',t'}) = \infty$ if there is no such edge. Also let $F(v_{R',t'})$ be the total weight of the shortest path from vertex $v_{\{s\},t_0}$ to $v_{R',t'}$. We have the following recurrence relation:

$$F(v_{R',t'}) = \min_{v_{R,t}}(F(v_{R,t}) + W(v_{R,t}, v_{R',t'})),$$

where $R \subset R', t = t'$ or $R = R', t = t' - 1$, for $R' = \{1, \ldots, n\}$, and otherwise, $R \subseteq R', t = t' - 1$.

For boundaries, we have

$$F(v_{\{s\},t_0}) = 0$$

and

$$F(v_{R,t_0}) = \infty, \quad \text{for } R \neq \{s\}.$$

Given the relation and the boundary values, we can compute the weight of the shortest path from $v_{\{s\},t_0}$ to each vertex from top to bottom and, for each row, from left to right. The minimum outcome among the total weights to the last-row vertices is, thus, our expected result. The corresponding shortest path (as well as the forwarding schedule that reliably broadcasts the message to all nodes) can be derived by a simple backtracking, and we refer to it as the *last row shortest path*.

Since the time dimension of the graph is infinite, there are potentially infinite number of paths to the last row. To overcome this problem, we introduce a terminating condition for each row, which guarantees that a shortest path to the last row can be found when the condition is satisfied for each row.

This condition is recursively defined as follows: a row indexed by R is *terminated* at time t, if

1. All the rows that have forwarding edges toward row R have terminated at some time t' before t; and
2. For any edge originated from a vertex of row R after t to a vertex of another row R', there must be at least one edge originated from a vertex of row R, of the same or less weight and in time $(t', t]$, to a vertex of row R'.

For boundary cases, we define that the first row satisfies the first condition from the very beginning, i.e., at time t_0, and the last row always satisfies the second condition, for there is no forwarding edge from it. We will then have the following theorem:

Algorithm OptimalForwardingSequence()

```
1:  F(v_{{s},t_0}) ← 0;
2:  t = t_0;
3:  while last row is not terminated,
4:    for ∀ row R not terminated
          and F(v_{R,t}) exists,
5:      for ∀ v_{R',t'} has an edge from v_{R,t},
6:        if F(v_{R',t'}) does not exist,
7:          F(v_{R',t'}) ← ∞;
8:        end if
9:        if F(v_{R,t}) + W(v_{R,t}, v_{R',t'})
              < F(v_{R',t'}),
10:         update F(v_{R',t'});
11:       end if
12:     end for
13:     if row R meets its terminating condition,
14:       terminate row R;
15:     end if
16:   end for
17:   t ← t + 1;
18: end while
19: find the minimum in last row;
20: return the last-row shortest path (correspond
        to the optimal forwarding sequence);
```

Fig. 3. The dynamic programming algorithm to compute the optimal forwarding sequence.

Theorem 4.1. *A shortest path to the last row is found when the last row is terminated.*

Proof. The proof is done by induction on the number of the forwarding edges used by a path to the last row. The key idea is that, for any path found after a row is terminated, we can always construct a path with less or equal total weight, where all its edges that pass the row are of the time before the row is terminated. The full proof can be found in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2011.94. □

The theorem directly leads to a dynamic programming algorithm, as shown in Fig. 3. Note that there are two strategies for calculating the recurrence relation: 1) starting from a vertex, find out those vertices that have edges to it; and 2) starting from a vertex, follow its edges and find out the vertices that these edges lead to. The second strategy is indeed much simpler and more efficient to implement. Specifically, it avoids unnecessary computations of those vertices with ∞ minimum costs, because no path from $v_{\{s\},t_0}$ really leads to them.

5 Distributed Solution: A Scalable and Robust Implementation

Using the centralized optimal algorithm, it is easy to evaluate the lower bound of the latency or the message forwarding cost to cover a given network, as well as the

trade-off between them. It, thus, offers a valuable benchmark to assess diverse broadcast strategies for duty-cycle-aware broadcast under ideal situations. It may also be practically useful for small networks with a centralized entity (e.g., the sink or base station) and for low-frequent broadcast of large messages, e.g., a code image update. For large-scale networks, however, the algorithm will suffer from the higher computation cost, and more importantly, from the increasing difficulty of obtaining the global connectivity and active/dormant patterns. The error-prone wireless communication raises additional challenges for information collecting and for reliable message forwarding.

In this section, we address these practical issues, and present a distributed scalable solution, which also well resists to wireless losses.

5.1 Generate Scalable Forwarding Sequence

For each sensor node, our distributed solution will focus on optimal forwarding sequence covering nodes within two hops, that is, the *1-hop neighbors* of the node and their neighbors, which we call *2-hop neighbors*. The reasons to choose two hops are threefold: first, it minimizes the computation overhead, and yet keeps reasonable accuracy; second, since every node must maintain information about its direct neighbors, the topology and active/dormant information for 1- and 2-hop neighbors can be obtained through a simple beacon protocol, without any extra broad-scope protocols for information dissemination; third, such information is sufficient to avoid most of message forwarding contentions, which will be further discussed in Section 7.

Assume that we are considering forwarding decisions at node w. We define a *Covering set*, or *CovSet*, as the set of 1- and 2-hop neighbors that are known (by w) being covered by at least one forwarding. A CovSet is created when a new broadcast message is received, and is updated when node w forward a broadcast message or a broadcast message is received or overheard. Specifically, when node w forward a broadcast message, based on the active/dormant patterns of its neighbors, it will find out those neighbors that are currently active and thus covered by this message, and then add them to its CovSet. And similarly, when a broadcast message is received or overheard, node w will also find out the currently active neighbors of the message's sender and add them to its CovSet. For example, in Fig. 1, if node 1 is active and nodes 0, 2, and 3 are also active, then after node 0 forward the broadcast message and node 1 overhears it, nodes 2 and 3 will be included in node 1's CovSet.

The CovSet is node w's view on the broadcast coverage states of its 1- and 2-hop neighbors. Accordingly, we modify the dynamic programming algorithm so that, for node w, it will calculate the forwarding sequence starting from the row of index equal to its CovSet. Also, the index of the last row will contain only the node w and its 1- and 2-hop neighbors.

Another challenge in distributed implementation is that the sequence calculation at different nodes are not necessarily synchronized, and are not even consistent, i.e., the forwarding sequence calculated by node w might not be followed by others that calculate their own sequences. To solve the inconsistency, when the CovSet is changed (updated), node w will check if this change follows its current forwarding sequence. For example, if the CovSet is changed due to an overheard message, node w then checks if this message is forwarded by the sender as indicated in its current forwarding sequence. If not, node w will recompute the forwarding sequence by incorporating the updated CovSet. Since the CovSet expands over time, the first row will become closer to the last row in each recomputation, implying that the computation cost reduces over time.

5.2 Accommodate Wireless Losses

The wireless channels are by nature error prone, and thus a neighbor might not successfully receive the message even if it is placed into the CovSet. For applications that require stringent coverage, we introduce a *Receiving Set*, or *RcvSet*, for each node w, as the set of 1- and 2-hop neighbors that are known (by w) having already received the message. Specifically, when node w receives a new broadcast message for the first time, it creates an RcvSet for this message and adds the sender of this message into the set. Afterward, when node w receives or overhears the broadcast message from its neighbor, it will add this neighbor into its RcvSet if this neighbor is not in the set yet. And when the RcvSet includes all its 1-hop neighbors, which guarantees that all its 1-hop neighbors have received the broadcast message (in spite of wireless losses), node w will affirm that no more message forwarding is necessary for itself and thus can safely stop.

To expedite the update of the RcvSet, when each node forward the message, it will piggy back its RcvSet by a bitmap. Its 1-hop neighbors, upon receiving or overhearing the message, will also update their RcvSet according to the piggy-backed RcvSet. Our simulation results suggest that this strategy significantly mitigates the impact of wireless losses, and by adding only a few explicit ACKs,[4] we can expect ideal (100 percent) reliability within given delay bounds.

Note that both RcvSet and CovSet are updated from node w's perspective, which might not reflect the real receving/covering status. In particular, the RcvSet might be a subset of CovSet only due to wireless losses. We use the CovSet in the forwarding sequence calculation, for it is an optimistic estimation and is thus more efficient. However, to prevent the CovSet from overexpanding that would adversely affect the efficiency, we will reset the CovSet to the RcvSet periodically, and also when the node turns active from the dormant state.

5.3 Summary

We summarize the core operations of the distributed solution in Fig. 4. When a node (e.g., node w) is in active state, it checks whether there is any arrived message (which can be either received or overheard). If so, node w will further process this message based on the message type. If the message is a new broadcast message, node w will create a RcvSet and CovSet for this message, and then add the

4. In our implementation, to accelerate the convergence of our solution, we also use a few explicit ACKs. For example, when node w receives a broadcast message only targeting on itself (which can be told by checking the piggy-backed RcvSet), it will send out an explicit ACK.

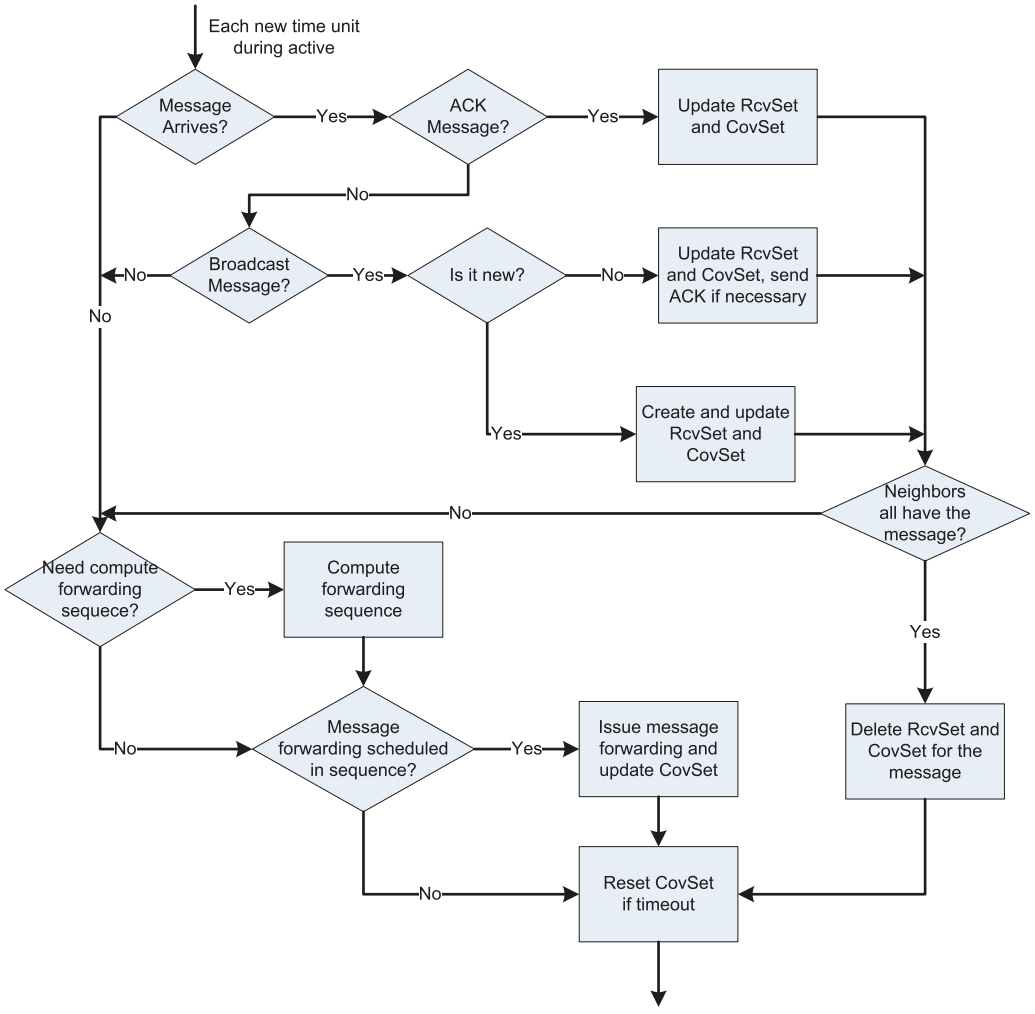

Fig. 4. Operations of the distributed solution (for an active node).

sender of this message and the neighbors in the RcvSet piggy backed with this message into its own RcvSet and CovSet. In addition, node w also adds its neighbors that are currently active and covered by this message into its own CovSet. If the arrived message is a broadcast message that has been received before, node w will then directly update the corresponding RcvSet and CovSet. It will also schedule to send an ACK if the arrived message is only targeting on itself. On the other hand, if the arrived message is an ACK, node w will update its RcvSet and CovSet by adding the sender of this message.

After processing the arrived messages, node w will then check its RcvSet whether all its neighbors have received the broadcast message. If so, node w can safely stop forwarding the broadcast message and release the memory used to store the corresponding RcvSet and CovSet. Otherwise, node w checks whether its CovSet updating follows its current forwarding sequence. If not, node w will further recompute its forwarding sequence. And if there is any message scheduled to forward, node w will send out the message. Finally, the CovSet will be reset to the RcvSet if there is a timeout.

In next section, we will show through simulations that our reliable broadcast solution is not only scalable and robust against wireless loss, but also retains near-optimal costs.

6 Performance Evaluation

In this section, we examine the performance of the proposed solution through extensive simulations. The following two major metrics are used in the evaluation: 1) Message cost, which is the number of forwardings (also reflecting the energy cost), and 2) Time cost, which is the total time slots taken to cover all the sensor nodes. We have examined diverse factors that impact the performance of our solution, including the duty cycle, network size/density and wireless communication losses. In this section, we present the results based on the following typical configurations, which are mainly adopted from [7], [9], [11], [20]. The sensing field is a square of 200 m by 200 m, and the wireless communication range is set to 10 m. The number of nodes in the network varies from 800 to 2,000. For each setting, we randomly generated 10 topologies. Each data point presented in this section is the average of 10 topologies with 10 runs on each topology. The active and dormant patterns are randomly generated following the duty-cycle value and exchanged among neighbors during the networking setup stage. We also adopt the wireless loss model used in [11], where packets are randomly dropped based on a predefined packet error probability.

As mentioned earlier, we do not assume any specific active/dormant schedule in our protocol design. Hence, we

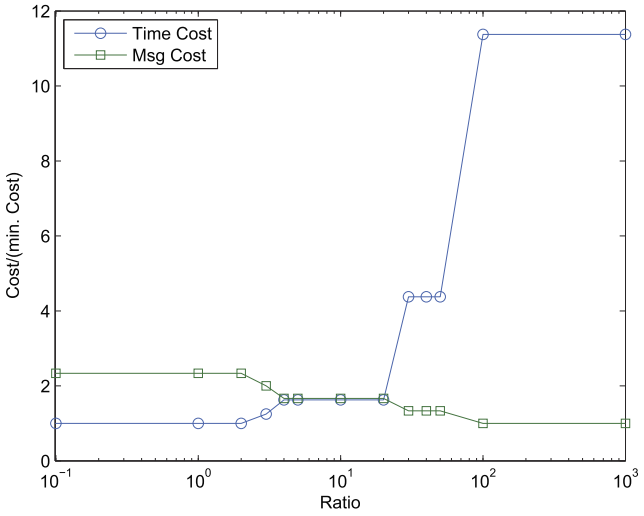

Fig. 5. The impact of the α/β ratio.

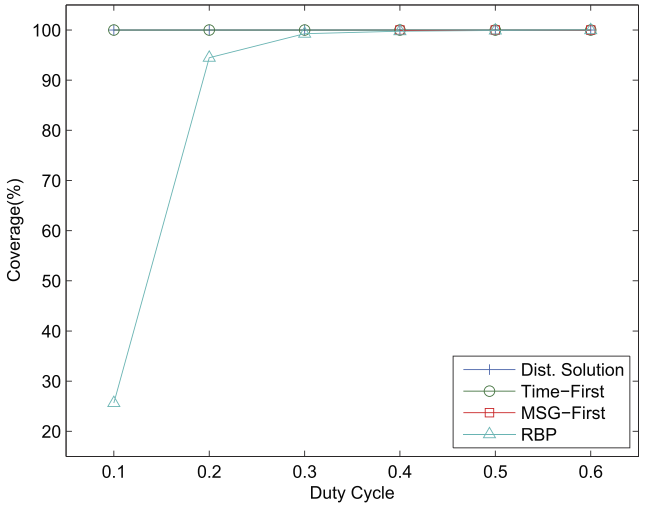

Fig. 6. Reliability under different duty cycles.

use various randomly generated schedules for performance evaluation and comparison.

6.1 Impact of the α/β Ratio

As mentioned, the optimal forwarding sequence depends on the ratio between α and β, while not their absolute values. We, therefore, first investigate the impact of this ratio. To minimize the influences from other uncertainties, we compute the time and message costs of different α/β ratios directly by the centralized solution, assuming the complete global knowledge is known. The results are shown in Fig. 5. For ease of comparison, the results are normalized by the respective minimum message and time costs. It is clear to see that, when α/β increases, the message cost decreases, while the time cost exhibits an inverse trend, in particular, it increases dramatically when the ratio exceeds 20.

A close look into the forwarding sequences generated by the centralized solution reveals that, most messages are forwarded by a node in one of the two phases: 1) when all (or almost all) of its noncovered 1-hop neighbors become active together; or 2) when the node or any of its noncovered 1-hop neighbors will turn dormant soon. This obviously can be locally determined, implying that our distribution solution could achieve good performance with no global information, which will be further validated in the following sections.

With no doubt, throughout the broadcast process, the share of 1 and 2 depends on α/β. An interesting observation comes from the region when α/β is around 10. In this region, both time and message costs remain unchanged with relatively low values. In other words, the objectives of minimizing time cost and message costs are pretty consistent in this region, which follows our intuition that, except for extreme cases, less message forwardings are accomplished in a shorter time. Hence, for our distributed solution, we use $\alpha/\beta = 10$ as the default setting, which enables a good trade-off and its results are well consistent with other settings within this region (see [23]).

To better understand the trade-off and for comparison, we also checked two extreme cases where α (or β) being equal to 0, and the other being greater than 0. These two settings simply lead to greedy strategies toward the lower bounds of the time cost and message cost, respectively, and we, thus, refer to them as *Time-First* and *Message-First* strategies. In practice, they can be implemented by purely using the aforementioned operation 1 (for Message-First) or 2 (for Time-First) with little modification. We have also embedded the same loss-recovery mechanisms as in our distributed solution.

6.2 Adaptability to Duty Cycle

We first examine the performance of our solution under different duty cycles, especially under low duty cycle. The network size is set to 2,000 nodes. Based on the values reported in [20] on observed link loss for real-world sensor nodes, the wireless loss rate is set to 0.3. For comparison, we also implemented RBP [20], a state-of-the-art broadcast algorithm for WSNs. RBP is flooding based with local repairs; it targets reliable broadcast but does not explicitly consider duty cycles. We found that when the duty cycle went low, this original RBP performed poorly and even failed to achieve the primary goal of reliability. The results are shown in Fig. 6 and detailed in Fig. 7 for duty cycle from 0.2 to 0.6. In contrast to the 100 percent reliability achieved

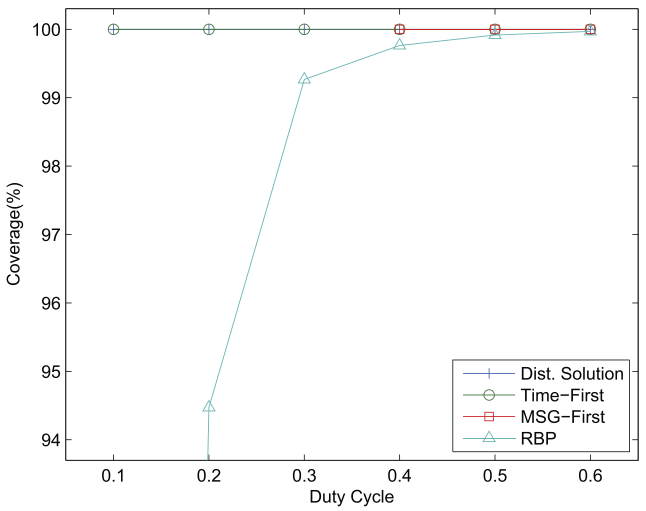

Fig. 7. Reliability under different duty cycles (amplified).

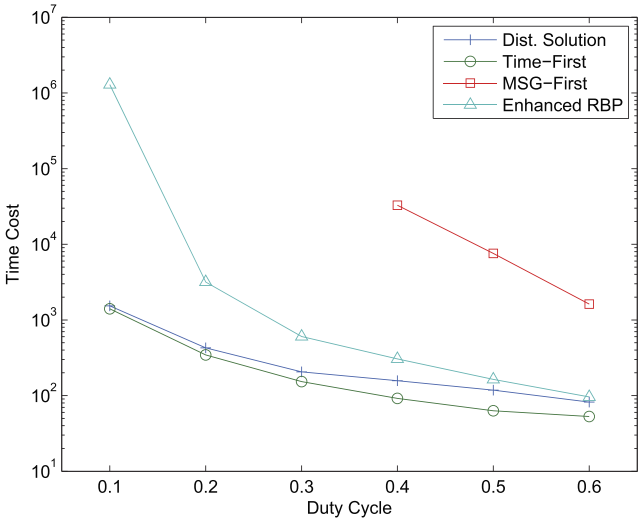

Fig. 8. Time cost under different duty cycles.

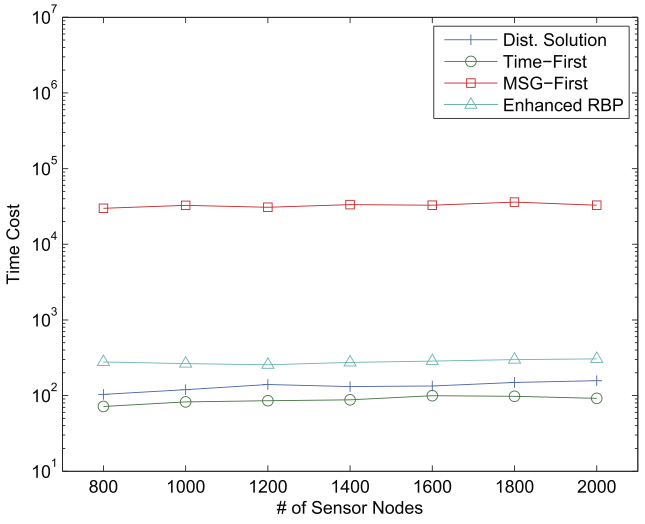

Fig. 10. Time cost with different number of sensor nodes.

by our distributed solution and the Time-/Message-First strategies, the reliability of RBP becomes unacceptable when the duty cycle is below 0.5.

It is also worth noting that, when the duty cycle is lower than 0.4, Message-First fails to terminate in finite time, because the probability for all noncovered neighbors being active simultaneously becomes extremely low. As such, it is not shown for these extremely low duty cycles. In fact, the strategy of Message-First, i.e., to wait until all noncovered neighbors become active, is our initial attempt to modify RBP to accommodate duty cycles. Unfortunately, our results suggest that this intuitive approach does not work in this new network context.

To achieve a fair comparison, we further enhance RBP by reissuing a broadcast immediately after the previous one until the required coverage reliability is achieved. This is motivated by the reliability compensation technique in [20]. The comparison results with this Enhanced RBP are shown in Figs. 8 and 9 for time and message costs, respectively. We can see that for both time and message costs, our distributed solution outperforms the Enhanced RBP and is close to the practical lower bounds of time and message costs given by the Time- and Message-First strategies, respectively. Moreover, although the Enhanced RBP performs close to our distributed solution under moderate duty cycles, say, 0.6, its performance degrades dramatically as the duty cycle becomes extremely low (note that the y-axis is in log-scale, and the x-axis is from low duty cycle to high). This further demonstrates the challenges caused by the invalidation of the all-node-active assumption and the necessity of designing a new broadcast service for low duty-cycle WSNs.

Compare with the Enhanced RBP and the Time-/Message-First strategies, our distributed solution is actually self-adaptive to different duty cycles. When the duty cycle increases, i.e., more opportunities to cover multiple neighbors with one forwarding, our solution successfully captures these opportunities and behaves like Message-First with very low message costs. On the other hand, when the duty cycle becomes extremely low, our solution does not waste time to blindly wait for such opportunities and performs more like Time-First to achieve low time costs.

In short, the invalidation of the all-node-active assumption renders the existing approaches (e.g., the original RBP) to be suboptimal or even failed under low duty cycles. Such extensions as the Enhanced RBP remain ineffective. On the other hand, our distributed solution adapts well to different duty cycles, with costs being close to the lower bounds of both time and message forwarding.

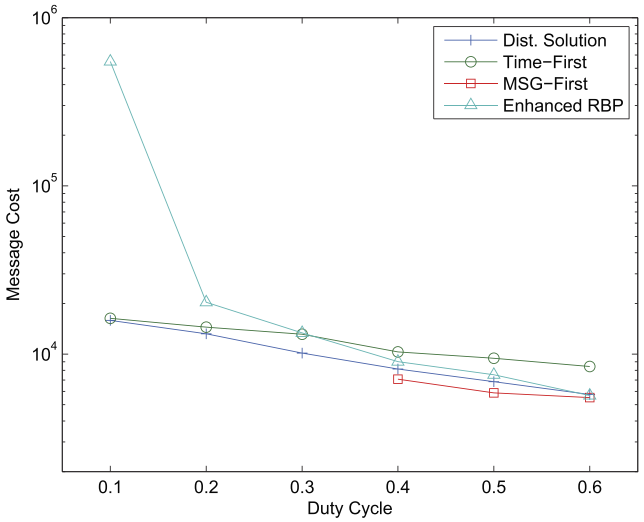

Fig. 9. Message cost under different duty cycles.

6.3 Scalability with Network Size

We next evaluate how the performance changes with different numbers of sensor nodes. We vary the number from 800 to 2,000 and the size of the sensing field is changed accordingly to keep the density. The impact of the different node densities will be investigated in the next section. Figs. 10 and 11 show the results for a default wireless loss rate of 0.3 and a duty cycle of 0.4. It is clear to see that the time cost of our solution is close to Time-First, and much less than those of the Enhanced RBP and Message-First. Meanwhile, the message cost of our solution is much less than those of the Enhanced RBP and

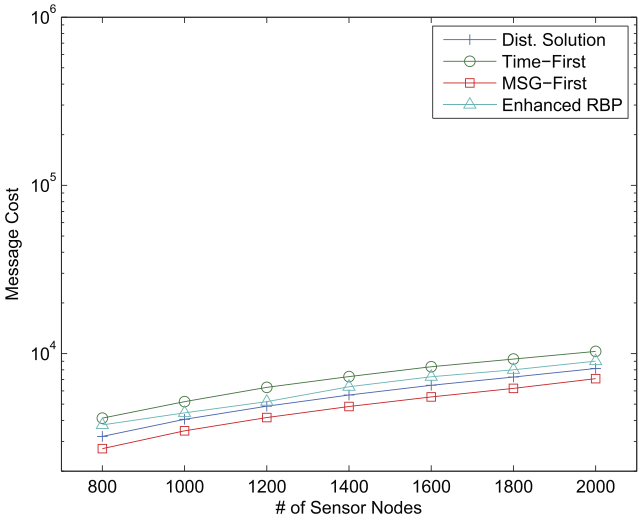

Fig. 11. Message cost with different number of sensor nodes.

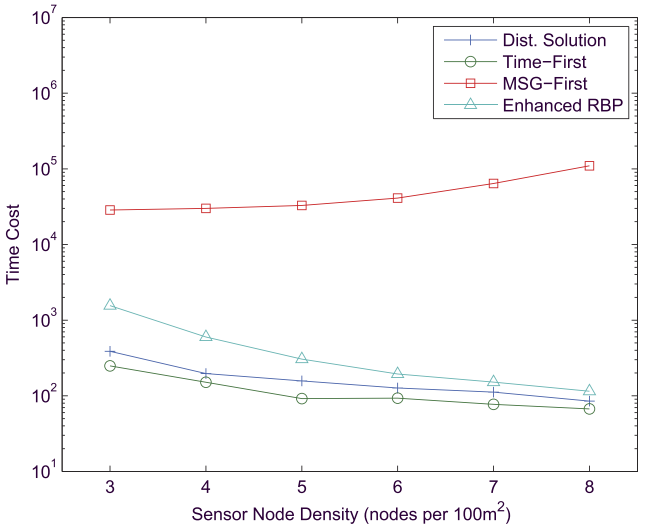

Fig. 12. Time cost with different sensor node densities.

Time-First, and is close to that of Message-First. This in turn justifies the existence of a stable region for α/β ratio selection, where both the time and message costs stay in relatively low values, in spite of the network size change.

With the number of nodes increases, our distributed solution, Time-First and the Enhanced RBP all exhibit a very slowly increasing trend for the total time consumed (see Fig. 10). This is because the time cost increment introduced by the increasing network diameter is not significant when comparing with the time costs caused by waiting for nodes to turn active. This is more notable for the Message-First approach, where the time costs spent by a node to wait for all its noncovered neighbors becoming active together dominate the total time costs and lead to an almost flat trend with only marginal variations.

For the total messages forwarded for broadcast, all four lines increase with the number of nodes, and the trend is almost linear (note it looks sublinear in Fig. 11 due to the log-scale of y-axis). However, our distributed solution grows relatively slower than both the Time-First and the Enhanced RBP, and stays close to the Message-First, which implies our distributed solution is scalable with the network size.

6.4 Effect of Network Density

We now examine the effect of different node densities in this section. To this end, we keep the node number as 2,000 and vary the size of the sensing field to change the node density from three nodes per $100\ \mathrm{m}^2$ to eight nodes per $100\ \mathrm{m}^2$. Figs. 12 and 13 give the results for a default wireless loss rate of 0.3 and a duty cycle of 0.4. It is easy to see that the time costs of our solution are very close to Time-First and much less than those of Message-First and the Enhanced RBP, in spite of the network density changes. Also, the message costs of our solution stay close to Message-First in most cases and are always less than the Time-First and the Enhanced RBP strategies.

An interesting observation is that as the node density increases, the time costs of the four approaches demonstrate different trends. Specifically, our solution, Time-First and the Enhanced RBP have lower time costs with higher densities while Message-First suffers higher time costs. The reason behind is that when the node density increases, a node will have more neighbors, which causes that a node has more chances to be covered by a message forwarding from its neighbors and thus reduces the time costs. On the other hand, in the Message-First strategy, having more neighbors implies that a node has to wait longer for its noncovered neighbors to become active together, and as a result, the total time costs are increased. Another observation is that the Enhanced RBP behaves differently for low and high node densities, which can be more clearly identified in Fig. 13, as its message costs below the density of five nodes per $100\ \mathrm{m}^2$ drop very quickly and then keep almost flat afterward. This is due to the local topology adaptation mechanism in RBP, which conducts local repairs by more retransmissions in low node density areas and less retransmissions in high node density areas. As the nodes are randomly deployed in our evaluation, with the average node density increasing, high node density areas become more prevalent and then dominate after the average density of five nodes per $100\ \mathrm{m}^2$.

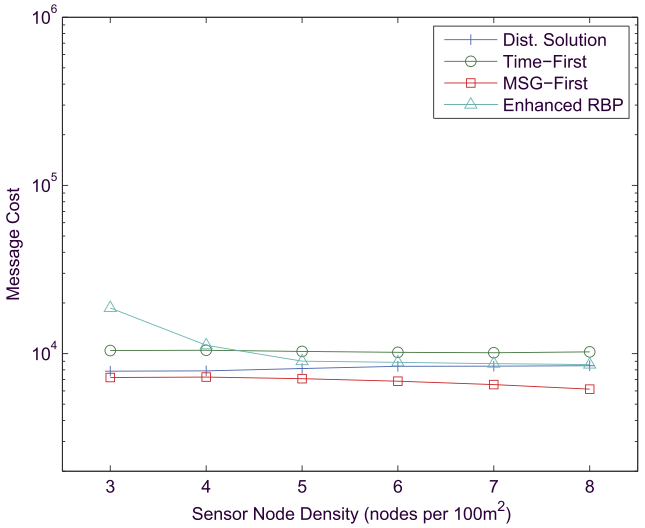

Fig. 13. Message cost with different sensor node densities.

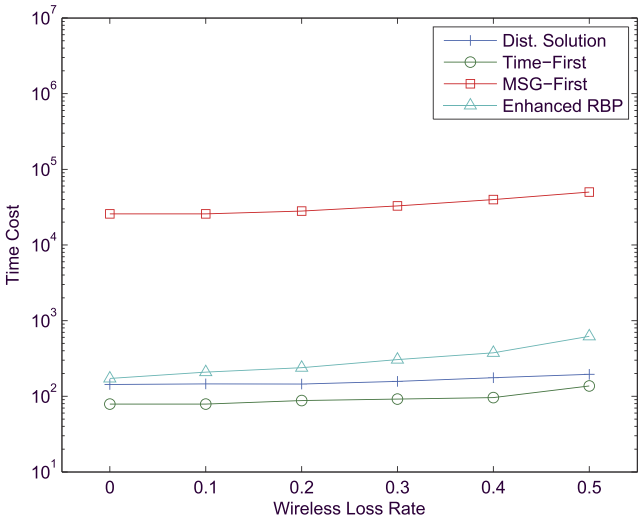

Fig. 14. Time cost under different wireless loss rates.

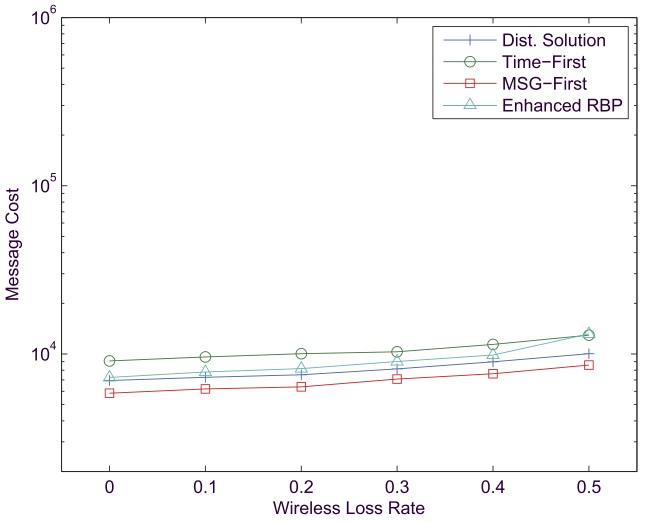

Fig. 15. Message cost under different wireless loss rates.

6.5 Robustness against Wireless Loss

We also investigate the impact of the wireless communication loss rate. Figs. 14 and 15 show the results for network size of 2,000 nodes and duty cycle of 0.4. Again, for all different loss rates, we find that, in terms of time cost, our distributed solution outperforms both Message-First and the Enhanced RBP, and is close to the Time-First. Regarding the number of message forwardings, our distributed solution is close to the Message-First and is much lower than both the Time-First and the Enhanced RBP.

Furthermore, Fig. 14 shows that the time cost of the Message-First increases faster (note that y-axis is in log-scale) when the wireless loss rate is greater than 0.1. This is because for the Message-First strategy, if a forwarded message gets lost, it takes a long time for a node to wait for all its noncovered neighbors from becoming active again and then have another forwarding. With the wireless loss increased, both the occurrence probability of such situations and the number of tries raise sharply. Similar situations also happen in the Enhanced RBP, where when a node misses a broadcast due to wireless loss, it may take quite a few time for next reissued broadcast to arrive at this node. On the other hand, the time costs of the Time-First and our distributed solution increase very slowly when the wireless loss rate is less than 0.4. This is because in low duty-cycle environment, the broadcast messages often have to be forwarded multiple times so as to cover all the nodes in an area. Due to the broadcast nature of wireless communications, a node may receive or overhear more than one broadcast message forwarding during its active periods. Thus, when the wireless loss rate is low, even a node misses the first message forwarding due to wireless losses, it still has a good chance to be timely covered by upcoming forwardings (though these forwardings are scheduled to cover other nodes in the area and not for this node). However, to cover all the nodes in an area, the Time-First introduces much more message costs, while our distributed solution successfully balances the time and message costs. It reacts to a loss increase with only a small number of extra message forwardings, and achieves a much lower time cost.

7 Further Discussion

In summary, our distributed solution achieves a near-optimal performance: its time cost is very close to that of the Time-First strategy, the lower bound of the time cost; and the message cost is very close to that of the Message-First strategy, the lower bound of the message cost. Yet, given that it involves local operations only, its computation and control overheads both scale well with the network size and density. It is also robust against wireless losses and cope well with different duty cycles.

It is worth noting that the broadcast module is intended to serve diverse applications. Hence, its control information could be piggy-backed or be overheard during the data transmissions of the served applications. This cross-layer optimization will further reduce the cost and enhance the responsiveness of our solution. Next, we briefly discuss two additional practical concerns for deploying our solution.

Collision reduction. Collision frequently happens in multi-hop wireless networks, particularly during data bursts, such as broadcast. For example, in Fig. 1, if nodes 1 and 4 forward messages to their neighbors simultaneously, then node 0 may get nothing because the two forwardings collide with each other. Existing broadcast algorithms passively rely on the MAC layer to avoid or resolve collisions, e.g., [18], [17], [19], which is indirect and thus can be inefficient. Our solution, however, can proactively detect the forwarding edges that potentially lead to collisions and remove them.[5] Furthermore, the asynchronous active/dormant patterns inherently reduce the chances of collision in low duty-cycle networks. We have verified this in our experiments, and have found that the collision probability becomes orders of magnitude lower than that in all-node-active networks. It, thus, becomes a less significant problem.

Flexible reliability. So far, we have struck to achieve the perfect reliability for broadcast. This is not mandatory in

5. For the collisions of ACKs, since only a few explicit ACKs are used in very specific cases, we consider the collision probability would be marginal. And even an ACK collision happens, our solution can still work well as after the collision, another broadcast message only targeting on a node will be forwarded later, which will trigger a second ACK to be sent out.

many applications, which seek better trade-offs among reliability, efficiency, and delay. Our solution can be easily modified to explore such flexibility. Specifically, it can terminate after all the neighbors not in RcvSet have been covered at least k times, where k is an application-controlled parameter, or to terminate when x percent neighbors of a node are in its RcvSet, where x can be determined based on local topology information, e.g., by approaches proposed in [20]. Our experience has shown that, by setting the coverage to 99 percent of the nodes, the time cost can be reduced by 50 to 75 percent in many low duty-cycle network [23]. This is because, in such networks, a small portion of nodes with low degrees would have low probabilities to activate together with their neighbors, and consequently a huge time margin has to be spent to cover these nodes.

8 Conclusion

In this paper, we revisited the broadcast problem in wireless sensor network with active/dormant cycles. We demonstrated that, under low duty cycles, conventional broadcast strategies assuming all-node-active would either suffer from poor performance or simply fail to cover the network. We took the initiative to remodel the broadcast problem in this new context. We showed that it is equivalent to a shortest path problem in a time-coverage graph, and accordingly presented an optimal centralized solution. This solution has also motivated a distributed implementation that relies on local information and operations only. We examined the performance of our solution under diverse network configurations, and compared it with state-of-the-art solutions.

We are continuing enhancing the performance of our distributed solution. Besides, we would like to implement our solution in real sensor networks as a generic service, and conduct experiments to investigate its interactions with applications. We are also interested in using probabilistic methods to model the trade-off between time, message costs, and reliability, thus extending our solution to support customized QoS demands from various applications. The use of our solution in delay tolerant networks (DTNs) [10], [13] is an another extension we are particularly interested in.

Acknowledgments

This work was supported by a Canada NSERC Strategic Grant, an NSERC Discovery Grant, an NSERC DAS grant, and an MITACS Project Grant. A preliminary version of this paper appeared in IEEE INFOCOM 2009.

References

[1] MICA2 Datasheet, http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2011.
[2] TinyOS Tutorial, http://www.tinyos.net/tinyos-1.x/doc/tutorial, 2011.
[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine*, vol. 40, no. 8, pp. 102-114, Aug. 2002.
[4] S. Deering, "Scalable Multicast Routing Protocol," PhD dissertation, Stanford Univ., 1989.
[5] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 784-803, Dec. 1997.
[6] Y. Gu, J. Hwang, T. He, and D.H.-C. Du, "uSense: A Unified Asymmetric Sensing Coverage Architecture for Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2007.
[7] Y. Gu and T. He, "Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2007.
[8] S. Guo, Y. Gu, B. Jiang, and T. He, "Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links," *Proc. ACM MobiCom*, 2009.
[9] X. Guo, "Broadcasting for Network Lifetime Maximization in Wireless Sensor Networks," *Proc. IEEE Comm. Soc. Conf. Sensor and Ad Hoc Comm. and Networks (SECON)*, 2004.
[10] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," *Proc. ACM SIGCOMM*, 2004.
[11] P. Kyasanur, R.R. Choudhury, and I. Gupta, "Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks," *Proc. IEEE Int'l Conf. Mobile Adhoc and Sensor Systems (MASS)*, 2006.
[12] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," *Proc. First Conf. Symp. Networked Systems Design and Implementation (NSDI)*, 2004.
[13] C. Liu and J. Wu, "Scalable Routing in Delay Tolerant Networks," *Proc. ACM MobiHoc*, 2007.
[14] J. Liu, F. Zhao, P. Cheung, and L. Guibas, "Apply Geometric Duality to Energy-Efficient Non-Local Phenomenon Awareness Using Sensor Networks," *IEEE Wireless Comm.*, vol. 11, no. 6, pp. 62-68, Dec. 2004.
[15] M. Miller, C. Sengul, and I. Gupta, "Exploring the Energy-Latency Tradeoff for Broadcasts in Energy-Saving Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2005.
[16] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," *Proc. ACM MobiCom*, 1999.
[17] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.
[18] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2003.
[19] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: A Hybrid MAC for Wireless Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2005.
[20] F. Stann, J. Heidemann, R. Shroff, and M.Z. Murtaza, "RBP: Robust Broadcast Propagation in Wireless Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2006.
[21] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D.B. Johnson, "ADB: An Efficient Multihop Broadcast Protocol Based on Asynchronous Duty-Cycling in Wireless Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys)*, 2009.
[22] Y. Sun, O. Gurewitz, and D.B. Johnson, "RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks," *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys)*, 2008.
[23] F. Wang and J. Liu, "On Reliable Broadcast in Large-Scale Low Duty-Cycle Wireless Sensor Networks," technical report, Simon Fraser Univ., 2007.
[24] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2003.
[25] T. Yan, T. He, and J. Stankovic, "Differentiated Surveillance Service for Sensor Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2003.

Feng Wang received both the bachelor's degree and master's degree in computer science and technology from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently working toward the PhD degree in the School of Computing Science at Simon Fraser University, conducting research on wireless sensor networks and peer-to-peer live streaming under the supervision of Prof. Jiangchuan Liu. His research interests include wireless sensor networks, peer-to-peer live streaming, and distributed computing. In the summer of 2006, he interned in the Wireless and Networking Group at Microsoft Research Asia, where he conducted research on peer-to-peer live video streaming. He also conducted research in the Department of Computing at Hong Kong Polytechnic University as a visiting PhD student in the spring of 2008 and the spring and fall of 2009, where his research was on data collections in wireless sensor networks. In the spring of 2010, he visited the Wireless Ad Hoc Networks Lab in the Institute of Software at the Chinese Academy of Sciences, where he worked on the testbed and information system design for wireless sensor networks. He was awarded the Tsinghua University Scholarship for Excellent Student in 1998, 2000, and 2001. At Simon Fraser University, he was awarded the SFU-CS Graduate Entrance Scholarship in 2005, the Graduate Fellowship in 2007, 2008, and 2009, and the President's PhD Research Stipend Award in 2011. He was also awarded the Chinese Government Scholarship for Outstanding Self-Financed Students Studying Abroad in the year of 2009-2010. He has been a student member of the IEEE and the IEEE Communications Society since 2007.

Jiangchuan Liu received the BEng degree (cum laude) from Tsinghua University, Beijing, China, in 1999, and the PhD degree from The Hong Kong University of Science and Technology in 2003, both in computer science. He was the recipient of a Microsoft Research Fellowship (2000), Hong Kong Young Scientist Award (2003), and Canada NSERC DAS Award (2009). He was a corecipient of the Best Student Paper Award of IWQoS 2008, the Best Paper Award (2009) of the IEEE ComSoc Multimedia Communications Technical Committee, and the Canada BCNet Broadband Challenge Winner Award in 2009. He is currently an associate professor in the School of Computing Science, Simon Fraser University, British Columbia, Canada, and was an assistant professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong from 2003 to 2004. His research interests include multimedia systems and networks, wireless ad hoc and sensor networks, and peer-to-peer and overlay networks. He is an associate editor of the *IEEE Transactions on Multimedia*, an editor of *IEEE Communications Surveys and Tutorials*, and an area editor of *Computer Communications*. He is the TPC vice chair for Information Systems of IEEE INFOCOM 2011. He is a senior member of the IEEE and a member of Sigma Xi.