Duty-Cycle-Aware Broadcast in Wireless Sensor Networks

Feng Wang Jiangchuan Liu

School of Computing Science
Simon Fraser University
British Columbia, Canada
Email: {fwa1, jcliu}@cs.sfu.ca

***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 timeframe. 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 network size and wireless loss increases. In addition, it enables flexible control toward the quality of broadcast coverage.

I. 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.

This work is supported by a Canada NSERC Discovery Grant, an NSERC Strategic Project Grant and a Nokia University Relation Initiative Grant.

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 [10][19].

The all-node-active assumption is valid for wired networks and for many conventional multi-hop 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][13][21][22]; 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 micro-controller 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 VI-B) have shown that, for ultra low duty-cycles, a conventional broadcast strategy would simply fail to cover all the nodes within an acceptable timeframe.

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 network size and wireless loss increases. In addition, it enables flexible control toward the quality of broadcast coverage.

The remainder of this paper is organized as follows. Section II lists the related work. In Section III, we re-formulate the broadcast problem in low duty-cycle WSNs. We introduce a centralized optimal solution in Section IV. It is then extended to a scalable and robust distributed implementation in Section V. In Section VI, we present extensive simulation results to evaluate the performance of our solution. We further discuss some key practical issues in Section VII, and concludes the paper in Section VIII.

II. Related Work

There have been numerous studies on broadcast in wired networks and in wireless ad hoc networks [4][5][15]. 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 [10], 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 [8], a timing heuristic named FDL (Forwarding-node Declaration Latency) 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 RBP (Robust Broadcast Propagation) [19], 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 [11]. 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.

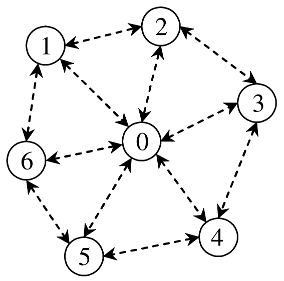

Fig. 1. An illustrative example for duty-cycle-aware broadcast. We use dashed lines for communications links, reflecting that they are not always available in the presence of duty-cycles.

Our work, different from 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 [14][7]. Among them, PBBF (Probability-Based Broadcast Forwarding) [14] offers an ultra-low duty-cycle MAC protocol with schedule channel polling. The MAC layer however considers operations of much shorter time-scales, and it does not involve many network-wide interactions, either. DSF (Dynamic Switch-based Forwarding) [7] considers data forwarding in low duty-cycles, but focuses on unicast from a data source to a sink. Our work complements them by considering the scenario of providing a network-wide broadcast service, which calls for novel solutions to ensure messages are successfully delivered to all destinations.

III. The Duty-Cycle-Aware Broadcast Problem

In this section, we re-formulate 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*. 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 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.

A. 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 loss[1], it is

[1]For ease of exposition, we do not consider wireless communication losses at this stage. Loss-tolerant mechanisms will be presented in Section V-B.

easy to find a simple schedule: Node 0 waits until all neighbors wake up and then forwards 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 forwards 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 6 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.

B. 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 i-th forwarding, where node u_i forwards the message at time t_i, and C_i be the set of nodes that receive the broadcast message in the i-th 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

$$S = \{(u_1, t_1), (u_2, t_2), \dots, (u_m, t_m)\} \quad (t_0 \le t_1 \le \dots \le t_m)$$

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 satisfies the following constraints:

(1) *Duty-cycle constraint*:

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

(2) *Forwarding order constraint*:

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

(3) *Coverage 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 coverage constraint ensures that the broadcast will cover all the nodes. This last constraint can be relaxed to achieve flexible reliability requirements, as will be discussed in Section VII.

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 non-urgent messages, such as a code update, a large α with a 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 VI-A.

IV. The Centralized Optimal Solution

We first transform the duty-cycle-aware broadcast problem into a shortest path problem in a time-coverage graph. Assume that the network topology and the active/dormant patterns are known, 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.

A. Time-Coverage Graph

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 $\{1, \dots, n\}$. Obviously, each

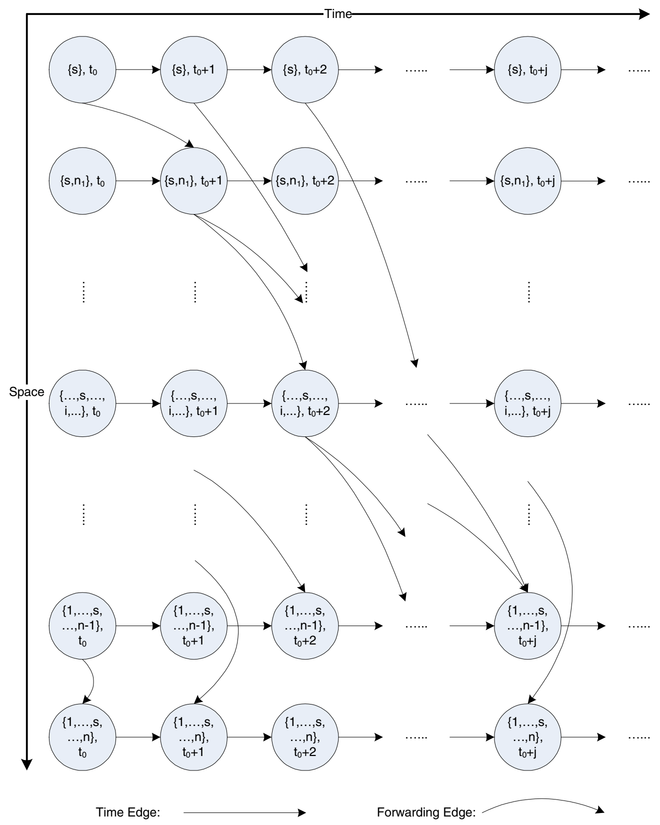

Fig. 2. An illustration of a constructed time-coverage graph.

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 R's 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 $t' = t$.

B. Equivalent Problem: Last-Row Shortest Path

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.

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

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

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 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.

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 [20].

```
Algorithm OptimalForwardingSequence()
1:  $F(v_{\{s\},t_0}) \leftarrow 0$;
2:  $t = t_0$;
3:  while last row is not terminated,
4:    for $\forall$ row $R$ not terminated and $F(v_{R,t})$ exists,
5:      for $\forall$ $v_{R',t'}$ has an edge from $v_{R,t}$,
6:        if $F(v_{R',t'})$ does not exist,
7:          $F(v_{R',t'}) \leftarrow \infty$;
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:   $t \leftarrow t + 1$;
18: end while
19: find the minimum in last row;
20: return the last-row shortest path (corresponding
21:         to the optimal forwarding sequence);
```

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

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 and 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.

V. Scalable and Robust Distributed 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. It is also 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.

A. Scalable Forwarding Sequence Generating

For each sensor node, our distributed solution will focus on optimal forwarding sequence covering nodes within 2 hops, that is, the *1-hop neighbors* of the node and their neighbors, which we call *2-hop neighbors*. The reasons to choose 2 hops are three-fold: 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 VII.

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. When node w forwards the message or when it overhears that a neighbor has done so, it will update its CovSet based on the local topology and active/dormant patterns. For example, in Fig. 1, if node 1 is active and nodes 0, 2 and 3 are also active, then after node 0 forwards 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 sequence. To solve the inconsistency, when node w remains active, we let it overhear the message copies forwarded by its 1-hop neighbors. If a neighbor does not follow the sequence, node w will re-compute 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 re-computation, implying that the the computation cost reduces over time.

B. Accommodating 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. When the RcvSet includes all its 1-hop neighbors, 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 forwards the message, it will piggy-back its RcvSet by a bitmap. Its 1-hop neighbors, upon receiving or overhearing the

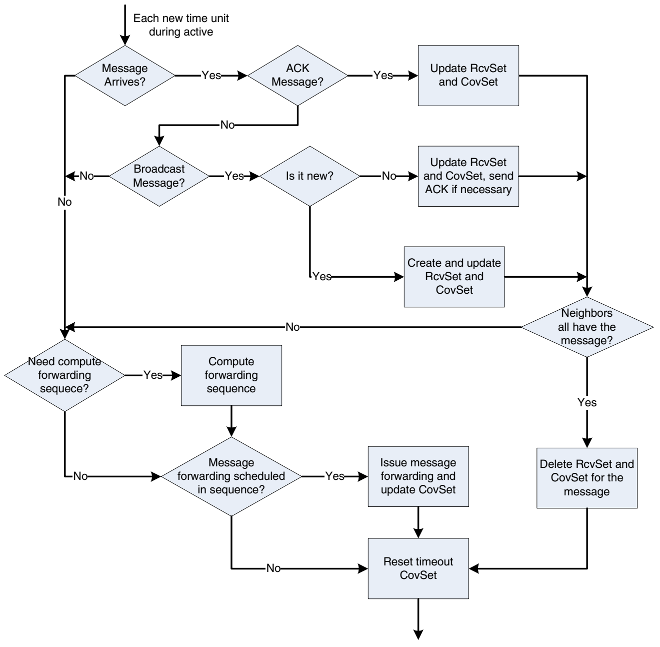
Each new time unit during active
Message Arrives?
Yes
ACK Message?
Yes
Update RcvSet and CovSet
No
Broadcast Message?
No
No
Yes
Is it new?
No
Update RcvSet and CovSet, send ACK if necessary
Yes
Create and update RcvSet and CovSet
Neighbors all have the message?
No
Need compute forwarding sequece?
Yes
Compute forwarding sequence
Yes
Message forwarding scheduled in sequence?
No
Yes
Issue message forwarding and update CovSet
Delete RcvSet and CovSet for the message
No
Reset timeout CovSet

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

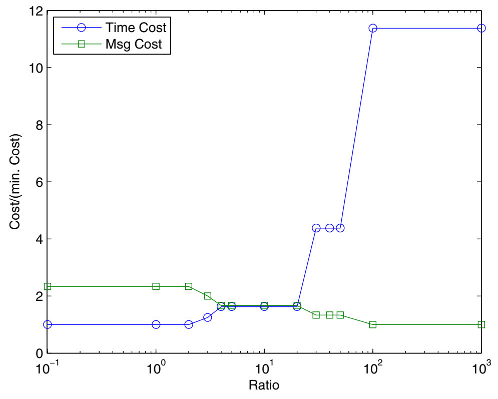

Fig. 5. The impact of the α/β ratio.

message, will update their RcvSet accordingly. Our simulation results suggest that this strategy significantly mitigates the impact of wireless losses, and by adding only a few explicit ACKs, we can expect ideal (100%) 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 over-expanding 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.

We summarize the core operations of the distributed solution in Fig. 4.

VI. 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 and wireless communication losses. In this section, we present the results based on the following typical configurations, which are mainly adopted from [7][8][10][19]. 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 2000, and for each number, 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 [10], 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 use various randomly generated schedules for performance evaluation and comparison.

A. *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 un-covered 1-hop neighbors become active together; or (2) when the node or any of its non-covered 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 subsections.

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

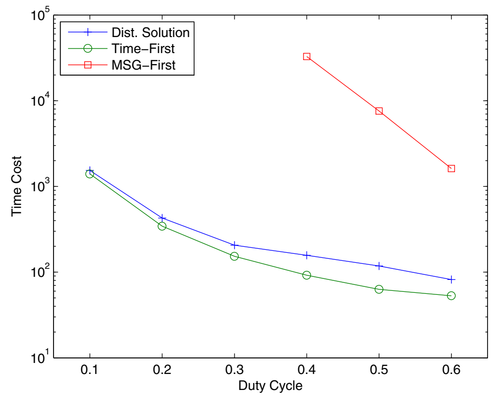

Fig. 6. Time cost under different duty cycles.

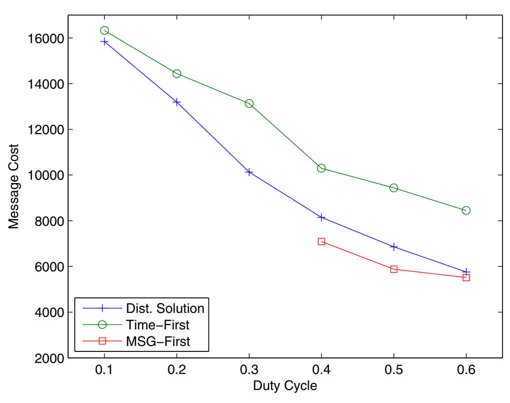

Fig. 7. Message cost under different duty cycles.

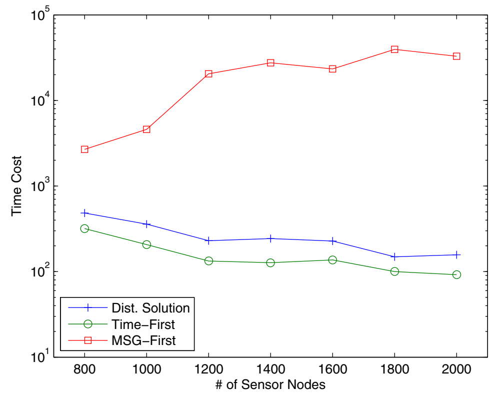

Fig. 8. Time cost with different number of sensor nodes.

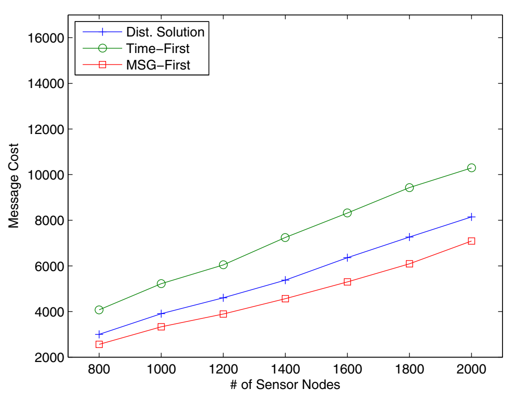

Fig. 9. Message cost with different number of sensor nodes.

message forwardings are accomplished in 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 [20]).

To better understand the tradeoff and for comparison, we also checked two extreme cases where α and β equal to 0 respectively, with 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.

B. 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 2000 nodes. Based on the values reported in [19] on observed link loss for real-world sensor nodes, the wireless loss rate is set to 0.3. The results are shown in Fig. 6 and Fig. 7. We can see that the time used by Message-First grows exponentially when the duty-cycle decreases (note that the y-axis is in log-scale, and the x-axis is from low duty-cycle to high). On the contrary, the time costs of our solution and Time-First increase much more slowly, and our solution performs very close to Time-First. Time-First however suffers from a high forwarding cost, while our solution does much better.

Note that, when the duty cycle is lower than 0.4, Message-First indeed fails to terminate in finite time, because the probability for all non-covered neighbors being active simultaneously becomes extremely low. As such, its costs for these low duty-cycles are not shown in these two figures. Recall that the design principle of Message-First is to wait until all non-covered neighbors become active, which, from a local viewpoint of each node, resembles all-node-active. It is thus not surprising that a broadcast strategy blindly based on all-node-active will easily fail in this new network context.

Also, an interesting finding is that our solution is actually self-adaptive to different duty-cycles. When the duty-cycle increases, which means more opportunities to cover multiple neighbors with one forwarding, our solution successfully captures these opportunities and behaves like Message-First with very low message cost. On the other hand, when the duty-cycle becomes extremely low, our solution does not waste time to

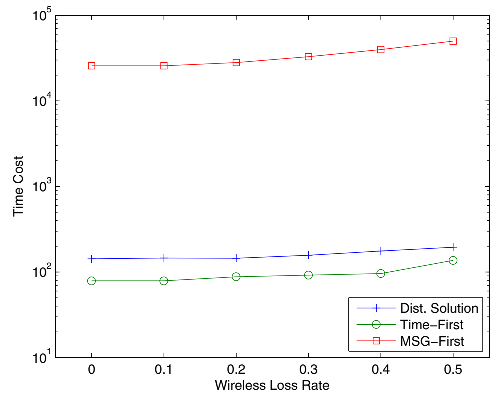

Fig. 10. Time cost under different wireless loss rates.

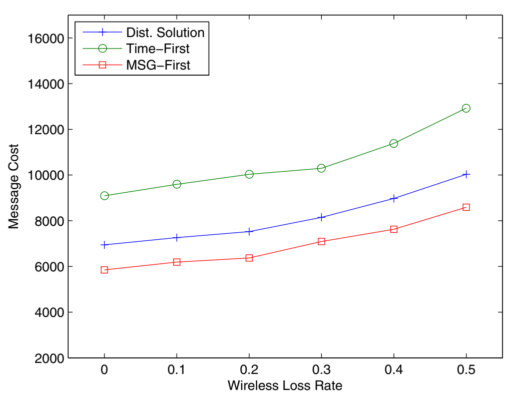

Fig. 11. Message cost under different wireless loss rates.

blindly wait for such opportunities and performs more like Time-First in achieving low time cost.

C. Scalability with Node Number

We next evaluate how the performance changes with different number of sensor nodes. Fig. 8 and Fig. 9 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 that of Message-First, particularly when the scale of the network grows. Meanwhile, the message cost of our solution is much less than that of 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 changes of network scales.

With the number of nodes increases, both our distributed solution and Time-First exhibit a decreasing trend for the total time consumed (see Fig. 8). This is because a node potentially has more neighbors when the number of nodes increases, which offers more chances for the node to be covered by forwarded messages during the same period. On the contrary, the time used for Message-First fluctuates and exhibits an overall increasing trend. We believe that it is because, given the asynchronous active/dormant patterns, for increased neighbors, a node has to spend more time to wait for all its non-covered neighbors from becoming active together.

For the total messages forwarded for broadcast, all three lines increase with the number of nodes, and the trend is almost linear. However, our distributed solution grows relatively slower than the Time-First and stays close to the Message-First, which implies our distributed solution is scalable with the network size.

D. Robustness against Wireless Loss

We also investigate the impact of the wireless communication loss rate. Fig. 10 and Fig. 11 show the results for network size of 2000 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 Message-First and is close to the Time-First. Regarding the the number of message forwardings, our distributed solution is close to the Message-First and is much lower than the Time-First.

Furthermore, Fig. 10 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 non-covered neighbors 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. On the other hand, our distributed solution successfully balances the time and message costs. It reacts to a loss increase with a small number of extra message forwardings, thus achieving a much lower time cost.

VII. 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 node 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., [17][16][18], which is indirect and thus can be inefficient. Our solution however can pro-actively detect the forwarding edges that potentially lead to collisions and remove them. Furthermore, the asynchronous active/dormant patterns inherently reduces 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 an less significant problem.

Flexible reliability: So far we have struck to achieve the perfect reliability for broadcast. This is not mandatory in 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 $x\%$ neighbors of a node are in its RcvSet, where x can be determined based on local topology information, e.g., by approaches proposed in [19]. Our experience has shown that, by setting the coverage to 99% of the nodes, the time cost can be reduced by 50% to 75% in many low duty-cycle network [20]. 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.

VIII. 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 tradeoff 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) [9][12] is an another extension we are particularly interested in.

References

[1] MICA2 Datasheet. [Online]. Available: http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

[2] TinyOS Tutorial. [Online]. Available: http://www.tinyos.net/tinyos-1.x/doc/tutorial/

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[4] S. Deering, "Scalable Multicast Routing Protocol," Ph.D. dissertation, Stanford University, 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 Transactions on Networking*, vol. 5, no. 6, pp. 784–803, 1997.

[6] Y. Gu, J. Hwang, T. He, and D. H.-C. Du, "μSense: A Unified Asymmetric Sensing Coverage Architecture for Wireless Sensor Networks," in *IEEE ICDCS*, 2007.

[7] Y. Gu and T. He, "Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links," in *ACM SenSys*, 2007.

[8] X. Guo, "Broadcasting for Network Lifetime Maximization in Wireless Sensor Networks," in *IEEE SECON*, 2004.

[9] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," in *ACM SIGCOMM*, 2004.

[10] P. Kyasanur, R. R. Choudhury, and I. Gupta, "Smart Gossip: An Adaptive Gossip-based Broadcasting Service for Sensor Networks," in *IEEE MASS*, 2006.

[11] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *USENIX NSDI*, 2004.

[12] C. Liu and J. Wu, "Scalable Routing in Delay Tolerant Networks," in *ACM MobiHoc*, 2007.

[13] J. Liu, F. Zhao, P. Cheung, and L. Guibas, "Apply Geometric Duality to Energy-efficient Non-local Phenomenon Awareness using Sensor Networks," *IEEE Wireless Communications*, vol. 11, no. 8, pp. 62–68, 2004.

[14] M. Miller, C. Sengul, and I. Gupta, "Exploring the Energy-Latency Tradeoff for Broadcasts in Energy-Saving Sensor Networks," in *IEEE ICDCS*, 2005.

[15] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," in *ACM MobiCom*, 1999.

[16] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Senor Networks," in *ACM SenSys*, 2004.

[17] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," in *ACM SenSys*, 2003.

[18] I. Rhee, A. Warrier, M. Aia, and J. Min, "Z-MAC: a Hybrid MAC for Wireless Sensor Networks," in *ACM SenSys*, 2005.

[19] F. Stann, J. Heidemann, R. Shroff, and M. Z. Murtaza, "RBP: Robust Broadcast Propagatin in Wireless Networks," in *ACM SenSys*, 2006.

[20] F. Wang and J. Liu, "On Reliable Broadcast in Large-Scale Low Duty-Cycle Wireless Sensor Networks," Simon Fraser Unviersity, Tech. Rep., 2007.

[21] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks," in *ACM SenSys*, 2003.

[22] T. Yan, T. He, and J. Stankovic, "Differentiated Surveillance Service for Sensor Networks," in *ACM SenSys*, 2003.