Fast Data Collection in Wireless Sensor Networks

Pradeep G[1]
[1] (P.G. Scholar,Department of Computer Science and Engineering, RVS College of Engineering and Technology, Coimbatore, India)
[1]pradeep.be2012@gmail.com

Abstract**:** *We study fast data collection in linear duty-cycled wireless sensor networks (WSNs). We first present a benchmark algorithm that can achieve optimality in a general case (i.e., in non-duty-cycled case). Then, based on the insights obtained in the general case, we propose an optimal algorithm and a distributed algorithm for the case when each sensor only works at one slot in a cycle in duty-cycled mode. It is proven that the two latter algorithms with cycle length equal to 3 have bounded performance gap to the benchmark algorithm in the non-duty-cycled case. Simulation results are used to demonstrate the effectiveness of the proposed algorithms*

Keywords: ***Data collection, duty cycle, sensor networks.***

I. INTRODUCTION

One essential mission of a wireless sensor network (WSN) is to efficiently deliver data collected by sensors to a base station (BS). In many WSN applications (e.g., the WSNs deployed along the roadside to detect car accidents, animals, and other dangerous road conditions), it is desired that the data be delivered as soon as possible, to minimize the completion time of data collection. The completion time may be affected by the following factors. First, a sensor is usually equipped with a half-duplex transceiver, which means it cannot send and receive simultaneously. Second, collisions may happen due to interference among sensors' transmissions.

Generally, minimum-time data collection in WSNs is an NP complete problem. Nevertheless, some optimal algorithms (noting that an optimal algorithm in this paper means that the algorithm minimizes the completion time of data collection) have been provided in the literature for some special cases. The work gives an optimal data collection algorithm for a linear WSN. The data collection problem is transformed into a data distribution problem, and then, the schedules in the data distribution problem are mapped into an optimal data collection solution. For a linear WSN in which each sensor has one packet to send, a distributed data collection algorithm is proposed, in which sensors rotate among three states ("transmission," "idle," and "receive"), and their initial states depend on their distance to the BS. Optimal data collection algorithms are given for a WSN with a tree topology, by assuming that interference is eliminated by power control or multichannel scheduling.

All above works assume that sensors are ready to receive or transmit packets at all times (i.e., they are in non-duty-cycled mode). However, usually, WSNs are expected to work for a long time (e.g., several months or even several years). If a sensor keeps running continuously, it may run out of battery in a few days. Therefore, a viable solution is to apply a duty-cycled mode. During each cycle of a number of slots, each sensor works in one or more working slots and sleeps in other slots. In particular, if each sensor is allowed to work at only one slot in a cycle, the working mode of the WSN is called low-power listening mode. Recently, duty-cycled WSNs have attracted much research attention. The major research efforts focus on routing and load balancing. To the best of our knowledge, there is no work in the literature that addresses fast data collection in duty-cycled WSNs considering the aforementioned two limiting factors (i.e., half duplex transceivers and interference).

To fill this research gap, in this paper, we investigate minimum-time data collection in duty-cycled WSNs with linear topologies, such as WSNs deployed along gas pipelines (to detect gas leakage), highways (to detect accidents), and underground tunnels (to detect gas and/or water), etc. To gain insights for designing data collection algorithms in duty-cycled WSNs, we first present a benchmark algorithm that is optimal in a more general case, i.e., in non-duty-cycled case. Then, based on insights obtained in the benchmark algorithm, we design an optimal algorithm and a distributed algorithm for duty-cycled case whose performance gap to the benchmark algorithm is bounded if the cycle length is equal to 3. Although the work also gives a centralized optimal solution for a non-duty-cycled linear WSN, the insights may not be able to be used in a duty-cycled WSN. This is because the scheme is based on the symmetry property of the data collection schedule (from sensors to the BS) and the data distribution schedule (from the BS to sensors). The symmetry property, which holds in a non-duty-cycled WSN, no longer holds for a duty-cycled WSN due to the fact that data transmission in a duty cycled WSN is receiver based (i.e., a node can receive packets only at its working slots, as shown in Section III). Noting that the optimal algorithm for linear non-duty-cycled WSNs is generally not unique, we solve the problem from a new perspective in this paper and provide insights that we may achieve optimality by giving higher priority to nodes closer to the BS and making simultaneous transmissions apart by at least three hops. As can be shown in Section III, these insights can be used in a duty-cycled WSN. Moreover, by using our proposed benchmark algorithm as a reference point, we show in Section III-B that our optimal duty-cycled algorithm and distributed duty-cycled algorithm have bounded performance gap to the optimal performance in the general case (i.e., in a non-duty-cycled WNS), whereas the performance bounds cannot be derived if we use the algorithm as a reference point for comparison.

II. BENCHMARK ALGORITHM IN NON-DUTY-CYCLED CASE

Consider a linear WSN as shown in Fig. 1. The network is denoted G = (V, E), with V = {0, 1, 2, . . . , N} being the set of nodes and E = {(i, i −1)|1 ≤i ≤N} being the set of wireless links. Node 0 is the BS, and node i is i hops away from the BS.

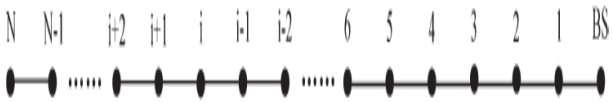

Fig. 1. Considered linear WSN.

Each node (e.g., node i) is equipped with an omnidirectional antenna and a half-duplex transceiver, and is able to communicate directly only with its two closest neighbors (node i + 1 on the left-hand side and node i – 1 on the right-hand side).1 Assume node i has wi(≥0) packets to be sent to the BS. For presentation simplicity, the packets are assigned with increasing indexes from node 1 to node N. The w1 packets of node 1 are indexed as 1, 2, . . . , w1, the w2 packets of node 2 are indexed as w1 + 1,w1 + 2, . . . , w1 + w2, and so on. Each node, e.g., node i, only transmits to its next-hop node, i.e., node i −1. Without loss of generality, when a node transmits to its next hop, it always sends the packet with the smallest index among all packets at its buffer (including its own packets and its received packets from other nodes). Therefore, during the data collection, it is impossible that a packet with a larger index is closer to the BS than a packet with a smaller index. Therefore, at the BS, packet 1 arrives first, whereas the packet with the largest index arrives last. Our target is an optimal data collection algorithm that has the minimum completion time for all the packets to arrive at the BS.

Similar to [1]–[4], time is divided into fixed-length slots, and in each time slot, multiple nodes may be scheduled to transmit simultaneously. Here, when we say a node is scheduled, it means the node can transmit
a packet.2 At a time slot, when node i is scheduled to transmit (to node i −1), nodes i + 1 and i −1 cannot be scheduled to transmit (to nodes i and i −2, respectively) due to the half-duplex transceiver, and nodes i + 2 and i −2 cannot be scheduled to transmit (to nodes i + 1 and i −3, respectively) due to the hidden terminal problem. In other words, among any three consecutive nodes, only one node can be scheduled to transmit at a time slot.

For a data collection algorithm, when the packet with the largest index arrives at node 3 at a time slot (e.g., slot t), all packets that have not arrived at the BS yet are with nodes 3, 2, and 1. Since only one node can transmit among these three nodes, it does not change the completion time if we arbitrarily schedule the remaining packets in time slots starting from slot t + 1. Since our target is the minimal completion time of gathering all packets, without loss of generality, we consider only data collection algorithms satisfying the following last-3-node assumption.When the packet with the largest index arrives at node 3 at a time slot, at any subsequent slot, the data collection algorithms always schedule the remaining packet with the smallest index to be sent.

Note that all assumptions related to the packet index are made to simplify presentation, and they are not mandatory for the proposed algorithms because our goal is to minimize the completion time of data collection. For example, if node i is scheduled to transmit by an algorithm, it can transmit any arbitrary packet in its buffer without changing the completion time of the algorithm.

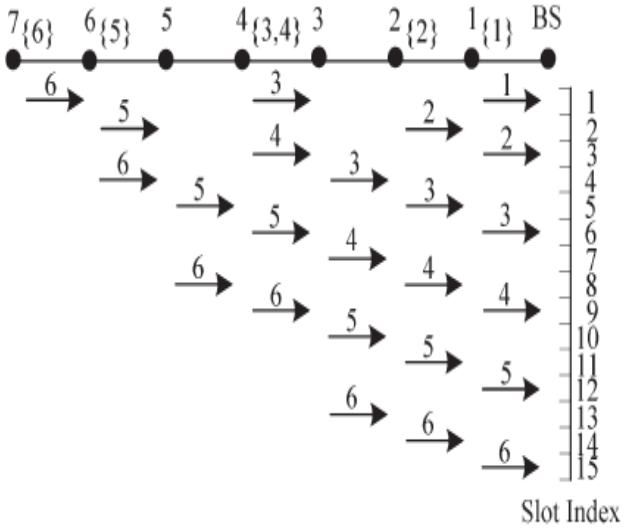

Fig. 2. Greedy data collection algorithm for a seven-node linear WSN.

Algorithm 1: Greedy data collection algorithm

```
1  begin
2      t ← 1;
3      while Buffer[0] < w_1 + w_2 + ... + w_N do
4          i ← 1;
5          while i ≤ N do
6              if Buffer[i] > 0 then
7                  Node i transmits a packet to Node i − 1
                   at Slot t;
8                  Buffer[i] ← Buffer[i] − 1;
9                  Buffer[i − 1] ← Buffer[i − 1] + 1;
10                 i ← i + 3;
11             else
12                 i ← i + 1;
13         t ← t + 1;
```

Since nodes closer to the BS have more traffic load, we propose Algorithm 1, which is a greedy algorithm that gives priority to those nodes. In Algorithm 1, Buffer[i] (i = 0, 1, . . . , N) denotes the number of packets at the buffer of node i. Therefore, at the beginning, Buffer[0] = 0, and Buffer[i] = wi, i = 1, 2, . . . , N. In the algorithm, t denotes the time slot index. At t = 1, we first schedule the node (e.g., node i) that has packets and is closest to the BS. Then, node i + 3 is checked (in line 10 of Algorithm 1) (recalling that, among any three consecutive nodes, only one node can be scheduled to transmit at a time slot). If the node has packets, then it is scheduled to transmit at the slot; otherwise, the node that is one-hop farther away from the BS is checked (in line 12 of Algorithm 1). This process is repeated until no more nodes can be scheduled, which completes the scheduling for t = 1. For t = 2, 3, . . ., the same procedure is repeated until all the packets arrive at the BS. Fig. 2 shows an example in a seven-node linear WSN. At the nodes, the numbers inside the brackets {· } are the indexes of packets generated at the corresponding nodes. Each arrow means a transmission, whereas the number above the arrow means the index of the packet that is transmitted. The

scale on the right represents the time slot scale. It can be seen that the data collection takes 15 time slots.

In data collection, if packet k is generated at node i, its transmission from node i to the BS can be described by time schedule (tki,/ tki−1, . . . , tk3, tk2, tk1), where tkj(j = i, i −1, . . . , 2, 1) is the index of the slot when node j transmits packet k. For instance, for the scheduling shown in Fig. 2, the time schedules of packets 1, 2, and 6 are (1), (2, 3), and (1, 4, 8, 9, 13, 14, 15), respectively.

Lemma 1: Consider a linear WSN with N nodes and K packets to be transmitted to the BS. Assume packet K is generated at node i. For an optimal data collection solution of the network, denote tK1 as the transmission moment (slot index) of packet K at node 1, which is also the minimum completion time of data collection of
the K packets. Suppose another packet, which is denoted as K + 1, is added at node j where j ≥i. For optimal data collection of the K + 1 packets, denote ˆtln (n = 1, 2, . . . , N; l = 1, 2, . . . , K + 1) as the index of the slot when node n transmits packet l. Then, ˆtK+1 1 is the minimum completion time of the data collection. We have three cases: 1) $\hat{t}^{K+1}1 = t^{K}_{1}+1$ if i = j = 1; 2) $\hat{t}^{K+1}_{1}= t^{K}_{1}+2$ if i = j = 2 ori = 1, j = 2; and 3) $\hat{t}^{K+1}_{1}\geq t^{K}_{1}+3$ if j ≥3 and j ≥i.

Proof: Nodes 3, 2, and 1 form a bottleneck of the data collection because, at any slot, only one node among them is allowed to transmit. Therefore, Cases 1 and 2 are obviously true. For Case 3, at an optimal data collection solution for the K + 1 packets, the time when node 3 transmits packet K + 1, which is denoted as $\hat{t}^{K+1}_{3}$, must be later than t^{K}_{1}because, otherwise, we can remove packet K + 1 from the transmission schedule and get a schedule for K packets that has a completion time less than t^{K}_{1}(recalling that, based on the last-3- node assumption, when packet K + 1 is sent at node 3, all other packets have already arrived at the BS). It takes two more slots for packet K + 1 to be sent from node 2 to the BS. Therefore, $\hat{t}^{K+1}_{1}=\hat{t}^{K+1}_{3}+2 \geq t^{K}_{1}+3$ if j ≥3 and j ≥i.

Theorem 1: Algorithm 1 is optimal.

Proof: We use mathematical induction for proving.

If there is only one packet to transmit, it is apparent that Algorithm 1 uses minimum time slots in data collection.

Suppose Algorithm 1 uses minimum time slots in data collection of any k packets in an N-node network. Now, we use Algorithm 1 to schedule k + 1 packets in the N-node network. We denote $\hat{t}^{l}_{n}$as the scheduling time of packet l(∈ {1, 2, . . . , k,k + 1}) at node n(∈{1, 2, . . . , N}). We denote t^{l}_{n}as the scheduling time of packet l(∈{1, 2, . . . , k}) at node n when Algorithm 1 is applied to schedule only packets 1, 2, . . ., k. Then, t^{k}_{1} is the minimum number of slots for data collection of packets 1, 2, . . ., k. Apparently $t^{l}_{n}= \hat{t}^{l}_{n}$, n ∈ {1, . . . , N},
l ∈ {1, . . . , k}. There are two cases as follows.

Case 1: Suppose packets k + 1 and k are both generated at node i, i >3 (for i ≤3, it is easy to prove).

Consider that Algorithm 1 is applied for the k + 1 packets. It can be seen that, at any moment, packet k + 1 is within three hops from packet k. Therefore, at the beginning of slot $\hat{t}^{k}_{3}$(i.e., the slot when packet k is sent by node 3 to node 2), packet k + 1 is with nodes 6, 5, 4, or 3, and all other packets (packets 1, 2, . . . , k −1) have already arrived at the BS. Then, Algorithm 1 takes the subsequent two slots, which are slots $\hat{t}^{k}_{2}=(\hat{t}k3+ 1)$ and $\hat{t}^{k}_{1}(= \hat{t}^{k}_{3}+ 2)$, to send packet k from node 2 to the BS. At the end of slot $\hat{t}^{k}_{1}$, packet k + 1 should be withnode 3. Since all other packets have arrived at the BS at this moment,Algorithm 1 takes three more slots to deliver packet k+1 from node 3 to the BS. In other words, $\hat{t}^{k+1}_{1}= \hat{t}^{k}_{1}+ 3$. Recall that $\hat{t}^{k}_{1}= t^{k}_{1}$(the minimum time slots needed for data collection of packets 1, 2, . . . , k). From Lemma 1, it can be concluded that $\hat{t}^{k+1}1$is the minimum number of slots for data collection of packets 1, 2, . . ., k + 1.

Case 2: Suppose packets k+1 and k are generated at nodes j and i, respectively, with j > i.

When j ≤i + 3, it means that packet k + 1 is initially within three hops from packet k. Then, when Algorithm 1 is used for the k + 1 packets, at any slot, packet k + 1 is always within three hops from packet k. Therefore, at the beginning of slot $\hat{t}^{k}_{3}$, packet k + 1 is with nodes 6, 5, 4, or 3. Similar to the proof in Case 1, it can be proven that $\hat{t}^{k+1}1$is the minimum number of slots for data collection of packets 1, 2, . . ., k + 1. Next, we consider the scenario when j > i + 3. Since there are no packets between nodes j and i, by Algorithm 1, packet k + 1 is transmitted in consecutive slots, starting from slot 1 until packet k + 1 is three hops away from packet k. If $\hat{t}^{k}_{3}$≤j −6 (i.e.,when packet k is scheduled at node 3, packet k + 1 cannot reach node 6), packet k + 1 is always more than three hops away from packet k; thus, packet k + 1 is scheduled continuously from slot 1 until it arrives at the BS at slot $\hat{t}^{k+1}_{1}$= j. Apparently, this completion time of datacollection of the k + 1 packets is minimum. If $\hat{t}^{k}_{3}$> j −6, it means that packet k + 1 can be within three hops away from packet k at a specific slot. Then, similar to the proof in Case 1, Algorithm 1 uses the minimum number of slots for data collection of the k + 1 packets.

The major drawback of Algorithm 1 is that it is a centralized algorithm and needs global information. Therefore, it is to be used to provide design insights and used as a comparison benchmark in Section III.

III. OPTIMAL AND DISTRIBUTED ALGORITHMS IN DUTY-CYCLED CASE

A. Optimal Algorithm in Duty-Cycled Case

In a duty-cycled scheduling, consider that each node (except the BS) is assigned one working slot in a cycle of T slots. During each cycle, a node (e.g., node i) wakes up at its working slot to receive a packet (if any) transmitted from node i + 1. Node i can also wake up to transmit a packet to its next hop neighbor, i.e., node i −1, at the working slot of node i −1. In other slots, node i is in sleeping state to save power. The BS is assumed to be in receiving state all the time. Similar assumptions are also adopted. Note that the transmission is receiver based: a feasible transmission from node i to node i −1 at slot t of a cycle must satisfy two conditions: 1) slot t is the working slot of node i −1; and 2) node I has packets to transmit. Denote the working slot of node i as τi ∈ {1, 2, . . . , T }, and assume T ≥3. The duty-cycled mode saves nodes energy butmay increase the completion time of data collection because a node only receives a packet at its working slot in each cycle. Recall that one insight in Algorithm 1 is that simultaneous transmissions should be apart by at least three hops. Therefore, the working slots of the nodes should be

assigned such that the working slots of any three consecutive nodes are different from each other. This can be done by a working slot assignment scheme when the WSN is initialized. To achieve this, we propose the following working slot assignment scheme for the initialization stage of a linear WSN with T ≥3. When the WSN is initialized, the BS sends a SLOT_ASSIGN packet, in which there is a field called WORKING_SLOT with initial value arbitrarily selected from {1, 2, . . . , T }. When a node receives the packet, it takes the value in theWORKING_SLOT field as its working slot index. The node then forwards the packet to the other side of the network but with the value in the WORKING_SLOT field reduced by 1. If the value equals 0 after the reduction, the value is set to T instead. By this working slot assignment scheme, the working slots of any T consecutive nodes are different from each other.

The following algorithm, which is denoted as Algorithm 2, is proposed.

1) Node i(2 ≤i ≤N) transmits at slot τi−1 if it has packets.

2) At any slot other than slots τ2 and τ1, node 1, if it has packets, transmits a packet to the BS.

3) At slot τ2, node 1, if it has packets, transmits when node 3 has no packet.

4) At slot τ1, node 1, if it has packets, transmits when node 2 has no packet.

In other words, in Algorithm 2, each node (except node 1) can transmit to its next-hop node at the working slot of the next-hop node, whereas node 1 can transmit to the BS at slots when nodes 3 and 2 do not transmit.

In Algorithm 2, node 1 is given more chances to transmit than other nodes. Therefore, Algorithm 2 is also greedy. We have the following theorem.

Theorem 2: Algorithm 2 uses a minimum number of slots in data collection in a linear WSN with a given working slot assignment in which each node, except the BS, is assigned with one working slot in a cycle, and the working slots of any three consecutive nodes are different from each other.

Proof: Consider node i, i ∈ {4, 5, . . . , N}. Since node i's receiving node, i.e., node i −1, has a different working slot from working slots of those nodes that are within two hops of node i −1, it is optimal for node i, if it has packets, to transmit a packet to node i −1 at the working slot of node i −1. Therefore, it is optimal to apply Part 1 of Algorithm 2 to nodes 4, 5, . . ., N. Next, we consider nodes 3, 2, and 1. The receiving node of nodes 3 and 2 are nodes 2 and 1, with working slot being τ2 and τ1, respectively. The receiving node of node 1 is the BS, which can receive at any slot. Then, apparently, it is optimal to apply Part 2 of Algorithm 2. Next, we prove that, for slot τ2, it is optimal to apply Part 1 of Algorithm 2 to node 3, and apply Part 3 of Algorithm 2 to node 1. In other words, we need to prove that, when both nodes 3 and 1 have packets at slot τ2, it is optimal to let node 3 transmit. We use proof by contradiction. Suppose it is not optimal to always schedule node 3 at slot τ2 when both nodes 3 and 1 have packets. Then, there exists an optimal schedule, which is denoted as O, such that, in slot τ2 of some cycles (e.g., L cycles denoted CL,CL−1, . . . , C1 ,where CL < CL−1 <· · · < C1), both nodes 1 and 3 have packets, but node 1 is scheduled. We have the following modification to O. Starting from cycle C1 + 1, find the first cycle, which is denoted cycle c, in slot τ2 of which node 3 transmits and does not have packet left in its buffer after the transmission; then, let node 3 transmit at slot τ2 of cycle C1, and let node 1 transmit at slot τ2 of cycle c. The resulting schedule, which is denoted as O1, is a feasible schedule and has the same completion time as that of O.

Similarly, we modify O1 such that node 3 transmits in cycle C2, whereas node 1 transmits in a later cycle. The resulting schedule, which is denoted O2, is a feasible schedule and has the same completion time as that of O1. This procedure is repeated for cycles C3, . . . , CL; eventually, we get schedule OL, which is a feasible schedule and has the same completion time as that of O. Thus, it is also optimal. The optimality of schedule OL contradicts our assumption (i.e., it is not optimal to always schedule node 3 at slot τ2 when both nodes 3 and 1
have packets) since schedule OL always lets node 3 transmit at slot τ2 when both nodes 3 and 1 have packets.

Similarly, we can prove that, when both nodes 2 and 1 have packets at slot τ1, it is optimal to let node 2 transmit.

Theorem 2 indicates that Algorithm 2 is optimal for duty-cycled scheduling. Since duty-cycled scheduling can be viewed as a special case of non-duty-cycled scheduling, the completion time ofAlgorithm 2 should be not less than that of Algorithm 1. However, in some special cases (e.g., when nodes have the same number of packets to be sent, which may happen when nodes periodically report their states for network diagnosis or respond to the query of the BS), the two algorithms have the same completion time, as shown in the subsequent Theorems 3 and 4. In the sequel, for presentation simplicity, we assume all algorithms in duty-cycled mode (i.e., Algorithms 2 and 3) use the proposed working slot assignment scheme to assign working slots to nodes.

Theorem 3: When there are N nodes and a BS in a linear dutycycled WSN, and each node has w packets to transmit to the BS, the completion time of Algorithm 2 is (N −1)w + 1 cycles if τ1 = T or (N −1)w cycles if τ1 < T.

Proof: At each of the first w cycles, each node other than nodes N and 1 sends a packet and receives a packet, whereas node N sends a packet, and node 1 receives a packet and can send at most T – 2 packets as it has T −2 "spare" slots (those other than slots τ1 and τ2). Therefore, at the end of cycle w, node N has no packet. Similarly, at the end of cycle 2w, node N −1 has no packet and so on; at the end of cycle (N −2)w, node 3 has no packet, node 2 has w packets, and node 1 has at most w packets (node 1 has w packets if T = 3 or one
packet ifT >3).

From cycle (N −2)w + 1 to cycle (N −1)w, node 1 can transmit in at least two slots in each cycle. If τ1 = T (i.e., the working slot of node 1 is at the end of a cycle), at the end of cycle (N −1)w, node 2 has no packet and node 1 has one packet. Then, Algorithm 2 takes one more cycle to deliver the packet in node 1 to the BS. Therefore, the completion time of data collection is (N −1)w + 1 cycles. If τ1 < T, at the end of cycle (N −1)w, nodes 2 and 1 both have no packet. Therefore, the completion time is (N −1)w cycles.

Theorem 4: For a linear WSN where each node (except the BS) has w packets, the completion time of Algorithm 2 with T = 3 and $\tau 1 < 3$ is the same as that of Algorithm 1.

Proof: Denote the number of nodes (except the BS) as N. For Algorithm 2 with T = 3 and $\tau 1 < 3$, the completion time is $3(N-1)w$ slots, according to Theorem 3.

Next, we assume N is a multiple of 3. The scenarios when N is not a multiple of 3 can be treated similarly.

For Algorithm 1, we have the following:

1) At the end of slot w, the numbers of packets at nodes 1, 2, . . . , N are 0,w, 2w, 0,w, 2w, . . . , 0,w, 2w, 0,w,w, respectively.

2) At the end of slot 3w, the numbers of packets at nodes 1, 2, . . . , N are 0, 0, 3w, 0, 0, 3w, . . . , 0, 0, 3w, 0, 0,w, respectively, and there are a total of $(N-2)w$ packets remaining at the nodes.

3) Starting from slot 3w + 1, Algorithm 1 delivers a packet to the BS after every three slots.

Therefore, the completion time of Algorithm 1 is $3w + 3 \times (N-2)w = 3(N-1)w$ slots, the same as that of Algorithm 2.

For a more general scenario (i.e., nodes have different numbers of packets), the difference of completion time of Algorithm 2 with T = 3 and Algorithm 1 is bounded, as will be shown in Section III-B.

B. Distributed Algorithm in Duty-Cycled Case

To implement Algorithm 2 in a distributed manner, the major challenge is that node 1 needs to know whether nodes 3 and 2 have packets to send so that it can "steal" unused slots by nodes 3 and 2, which is hard to achieve in a distributed manner. Accordingly, we let node 1 only steal unused slots by node 2 as follows. At the very beginning of slot $\tau 1$, when node 1 wakes up and senses the channel to be idle (which means node 2 is not transmitting), then node 1 transmits at slot $\tau 1$. The resulted distributed algorithm is called Algorithm 3. Since the channel sensing time can be very short (e.g., in IEEE 802.11 medium access control standard, 20 μs is more than sufficient to detect transmission of other nodes, whereas typical time slot duration in WSNs is at least a few dozen milliseconds), this sensing overhead is negligible.

Define $f_a(w1,w2, \ldots, wN)$ as the completion time of Algorithm a (a is algorithm index) for a linear WSN with node i(i = 1, 2, . . . , N) originally having wi packets. The following theorem gives a performance
bound for Algorithm 3.

Theorem 5: For Algorithm 3 with T = 3 and with the proposed working slot assignment scheme, we have

$$f_3(w1,w2, \ldots, wN) \le f_1(w1,w2, \ldots, wN) + 3(w1 + w2) + 2. \quad (1)$$

Proof: Assume that, using the working slot assignment scheme, the working slots of the nodes are $(\tau 1, \tau 2, \ldots, \tau N)$.

We have

$$f_3(w1,w2, \ldots, wN) \le f_3(0, 0,w3,w4, \ldots, wN) + f_3(w1,w2, 0, 0, \ldots, 0) \le f_3(0, 0,w3,w4, \ldots, wN) + 3(w1 + w2). \quad (2)$$

Here, the second inequality is because, when only nodes 2 and 1 have packets, Algorithm 3's completion time (units: slots) is at most three times the number of packets in nodes 2 and 1.

Then, consider the following modification of Algorithm 1. Change Step 6 from "if Buffer[i] >0 then" to "if Buffer[i] >0 and node i does not transmit in the previous two slots (slots $t-1$ and $t-2$) then". The resulting algorithm is called Algorithm 4. It can be proven that, for a linear WSN in which nodes 2 and 1 do not have packets, Algorithm 4 is optimal. The proof is almost the same as that of Theorem 1 and is thus omitted here. Since Algorithm 1 is also optimal, we have $f_4(0, 0,w3,w4, \ldots, wN) = f_1(0, 0,w3,w4, \ldots, wN)$.

Consider a linear WSN with $(0, 0,w3,w4, \ldots, wN)$ as the vector of packets originally at the nodes. Consider that Algorithm 4 is used for the WSN. Then, in the schedule, each node transmits at most once in any three consecutive slots. We group every three slots into a cycle. In the schedule of Algorithm 4, for each transmission of node $i(\in \{1, 2, \ldots, N\})$, if the transmission time is not slot $\tau i-1$ of a cycle (here, $\tau 0 \in \{1, 2, 3\} \setminus \{\tau 2, \tau 1\}$), then we postpone the transmission time to the next slot $\tau i-1$ (which should be either in the current cycle or in the next cycle). Therefore, each transmission is postponed by at most two slots. The resulted transmission schedule is still a feasible schedule. In the postponed schedule, node i only transmits at slot $\tau i-1$. Thus, the postponed schedule's completion time is not less than the completion time of Algorithm 3. In other words, we have $f_3(0, 0,w3, \ldots, wN) \le f_4(0, 0,w3, \ldots, wN) + 2$ (recalling that each transmission is postponed by at most two slots).

Based on the above results and $f_1(0, 0,w3,w4, \ldots, wN) \le f_1(w1,w2, \ldots, wN)$, we have (1).

Theorem 5 indicates that the completion time difference between Algorithm 3 with T = 3 and Algorithm 1 is bounded by $3(w1 + w2) + 2$. Since Algorithm 3 is implemented in a duty-cycled mode, its completion time is not less than that of Algorithm 2, which is optimal in duty-cycled mode. Therefore, the completion time difference between Algorithm 2 with T = 3 and Algorithm 1 is also bounded by $3(w1 + w2) + 2$.

Generally, for Algorithm 3, a larger cycle length T can achieve more energy saving (since a node can be in sleeping state for longer time) but may increase the completion time. On the other hand, a smaller T can benefit the completion time but consumes more energy.

Next, we extend Algorithm 3 to a low duty-cycled case by using adaptive duty-cycle technique, to save energy and reduce completion time. Specifically, when the WSN is set up, an idle mode is used with a large cycle length Tidle, which is a multiple of 3. The working slots are assigned according to the proposed working slot assignment scheme. In the idle mode, once a node, e.g., node i with working slot τi, receives a packet (i.e., node i + 1 detects an event or has received packets from node i + 2), node i will forward the packet to the next
node (node $i-1$), and change to working mode with cycle length Twork = 3 and use working slot $((\tau i - 1) \bmod 3) + 1$.

Node i also sets up a timer (e.g., 2Tidle slots) for the working mode. If no further packet is received within the timer duration, then node i changes back to the idle mode with cycle length Tidle. When Tidle is a multiple of 3, it can be guaranteed that, during the switching between the idle mode and the working mode, there is no conflict in the working slots of any three consecutive nodes.

The adaptive duty-cycle technique can strike a good balance between the completion time of data collection and network lifetime. When there is no traffic, nodes work in the idle mode with a large cycle length Tidle to save energy. When there is traffic, the network changes to working mode with Twork = 3 to achieve fast data collection (recalling that completion time of Algorithm 3 with T = 3 is close to the minimum completion time, with a bounded difference, as indicated in Theorem 5), and after all traffic has been delivered, the network returns to the idle mode to save energy.

C. Discussion

There are still some issues when the distributed algorithm (Algorithm 3) is implemented in real applications. One issue is that time synchronization is needed among nodes to implement slotted transmissions. However, only much loose time synchronization is needed in our distributed algorithm. A node, e.g., node i, only needs to synchronize with its next hop, i.e., node i −1, at the initialization stage. Such time synchronization can be achieved by a time synchronization protocol such as the flooding time synchronization protocol (FTSP) in [8] with only a few messages (because of the linear nature of the considered WSNs). After initial synchronization, by the FTSP, nodes i and i −1 can have clock synchronization as accurate as 2.24 μs through a few packet exchanges between them every 15 min. Therefore, it can be seen that the cost of time synchronization is low in our distributed algorithm.

Another issue is unreliable wireless communication links, which may cause transmission failures and retransmissions will be needed. Our Algorithm 3 can still work with transmission failures, as follows. In a linear WSN in which node i(i = 1, 2, . . . , N) originally has wi packets to be sent to the BS, the total number of different packets that should be transmitted by node i is_N_l=i wl. Considering transmission failures, for_N_l=i $w_l$different packets, denote qi as the expected total number of node i's transmissions/retransmissions and denote vi = qi −_N_l=i wl ≥0. Then, the problem is equivalent to data collection in a linear WSN in which node i(i = 1, 2, . . . , N) originally has wi +vi packets to be sent to the BS and all transmissions are successful.

Then, the objective of our algorithm is to minimize the expected data collection time. One shortcoming of linear topologies is that any node or link failure (which cannot be solved by retransmissions, such as a hardware problem or battery depletion) may disconnect the network. However, as previously mentioned, there are many applications in which the WSNs have linear topologies, and it is not cost effective to deploy a dense mesh network. To solve the node/link failure problem in a linear WSN, we can deploy some nodes as backup nodes, which will become active if node/link failures happen. Even if there is no backup node, nodes around the failed nodes can detect the link failure and take some actions to solve the problem (such as increase the transmission power).

IV. PERFORMANCE EVALUATION

We evaluate the proposed algorithms over a customized C/C++ simulator. The number of nodes N takes values from 50 to 100. Two typical traffic models are simulated. One is periodic reporting and the other is event detection. For periodic reporting, nodes report data to the BS every 30 min. In each reporting, for each node i(i ∈{1, 2, . . . , N}), wi is randomly chosen from 1 to 9. For event detection, assume events occur according to a Poisson process with rate λ = 2 per hour, which implies that event interarrival times have exponential distribution with a mean value of 30 min. When an event happens, wi randomly takes values from {0, 1, 2, 3} with probability {50%, 16.7%, 16.7%, 16.7%}. Unreliable wireless links are also simulated by setting

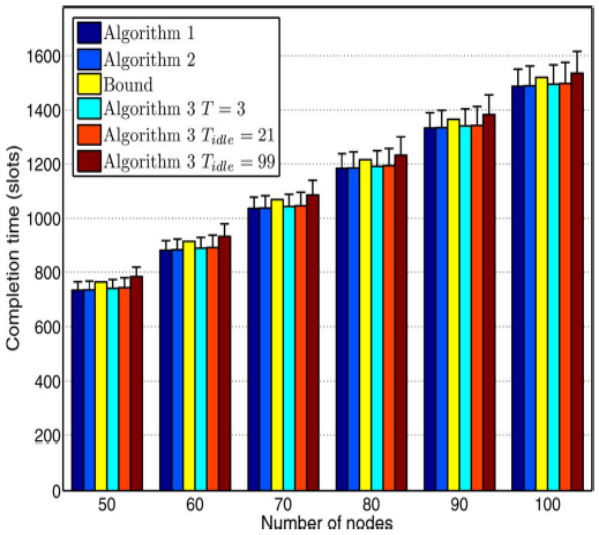

Fig. 3. Completion time of different algorithms for periodic reporting every 30 min.

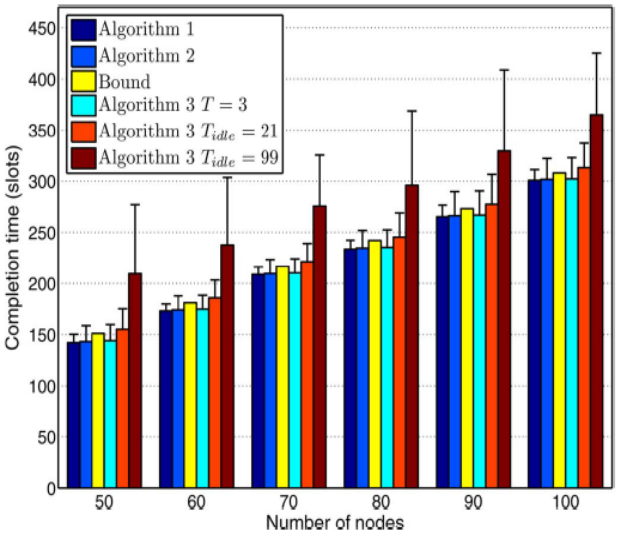

Fig. 4. Completion time of different algorithms for event detection with mean event interarrival time being 30 min.

the successful probability of a transmission/retransmission to be 0.8. The length of one slot is 20 ms, and the statistics are collected for 120 h of network time.

Figs. 3 and 4 show the average completion time of Algorithms 1 and 2 with T = 3, and Algorithm 3 with T = 3 and adaptive duty cycle (with Tidle = 21 and Tidle = 99, i.e., the initial duty cycle is approximately 5% and 1%, respectively), for periodic reporting and event detection, respectively. We also simulate the scheme in [2], and confirm that our Algorithm 1 and the scheme in [2] have the same completion time when the transmission links are reliable (although the schedules in the two algorithms are different). Therefore, we do not show the simulation results of the scheme in [2], and only Algorithm 1 is shown as the benchmark. The upper bound of completion time of Algorithm 3 with T = 3 (as indicated in Theorem 5) is also shown in the two figures. In the two figures, the histogram means the average completion time with reliable transmission links, whereas the line
segments above the histogram mean the completion time increase due to unreliable transmission links.

From the two figures, it can be seen that Algorithm 2 with T = 3 has slightly larger completion time than Algorithm 1, whereas Algorithm 3's completion time with T = 3 is very close to that of Algorithm 2. Compared with Algorithm 1, the completion time

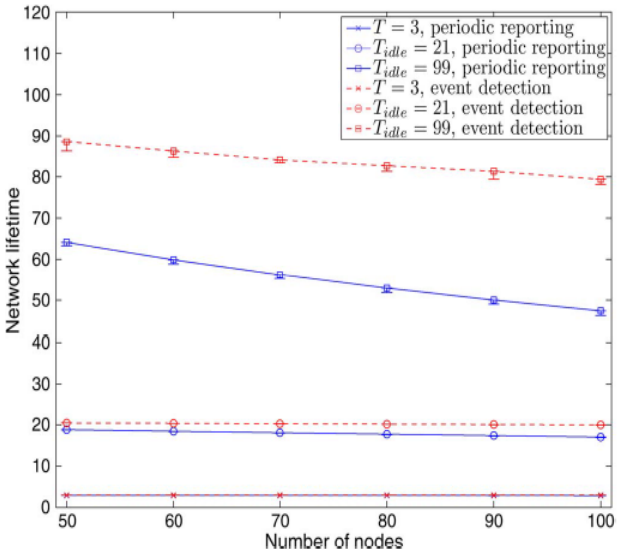

Fig. 5. Normalized network lifetime of Algorithm 3.

of Algorithm 3 with Tidle = 21 increases by no more than 2% (3% for unreliable links) for periodic reporting, and it increases by no more than 10% (17% for unreliable links) for event detection. Algorithm 3 with Tidle = 99 still performs well for periodic reporting, whereas its completion time increases apparently for event detection. The reason is as follows. For periodic reporting, as all nodes have packets to send, they can quickly switch to working mode. For event detection, the traffic is sparsely distributed along the linear network, and it takes some slots (e.g., half a cycle on average) for downstream nodes to change to working mode. In addition, it takes much more slots when transmission links are unreliable, as indicated by Fig. 4.

We next evaluate the network lifetime of Algorithm 3 with T = 3 and with an adaptive duty cycle. In each cycle, a node is in receiving state at its working slot, in transmitting state if it transmits at the working slot of its next-hop neighbor, or in sleeping state at other slots. According to [9], the power consumption for receiving and
transmitting in a WSN are similar, whereas the power consumption in sleeping state is very small and can be thus neglected. Therefore, in our simulation, we assume that the energy consumption of a node at a time slot is a constant positive value if the node is in transmitting or receiving state, and the energy consumption is zero when the node is in sleeping state. Since all packets have to go through node 1, we take the lifetime of node 1 as the network lifetime. We regard the lifetime of node 1 when it is always in receiving or transmitting state as one, and we measure the normalized network lifetime of our Algorithm 3 with T = 3 and with an adaptive duty cycle (with Tidle = 21 and Tidle = 99), as shown in Fig. 5. In the figure, the line segment under each discrete mark means the decrease in network lifetime due to unreliable transmission links. It can be seen that, for a given traffic model (periodic reporting or event detection), the network lifetime is approximately proportional to the cycle length. The network lifetime tends to decrease as the number of nodes in the network increases, as shown in Fig. 5. This is reasonable because, with more nodes, more traffic may be generated, which eventually needs to be transmitted by node 1.

V. CONCLUSION

For data collection in a linear WSN, we first provide an optimal solution for non-duty-cycled case as a benchmark. Then, we propose an optimal algorithm and a distributed algorithm for duty-cycled case. Our results will provide theoretical performance bounds for data collection time in linear WSNs. The insights gained in our proposed algorithms can be helpful for developing data collection algorithms for WSNs with other topologies.

REFERENCES

[1] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," Wireless Netw., vol. 16, no. 4, pp. 985–997, May 2010.
[2] C. Florens and R. McEliece, "Packets distribution algorithms for sensor networks," in Proc. IEEE INFOCOM, 2003, pp. 1063–1072.
[3] S. Gandham, Y. Zhang, and Q. Huang, "Distributed time-optimal scheduling for convergecast in wireless sensor networks," Comput. Netw., vol. 52, no. 3, pp. 610–629, Feb. 2008.
[4] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," IEEE Trans. Mobile Comput., vol. 11, no. 1, pp. 86–99, Jan. 2012.
[5] Y. Gu and T. He, "Dynamic switching-based data forwarding for lowduty- cycle wireless sensor networks," IEEE Trans. Mobile Comput., vol. 10, no. 12, pp. 1741–1754, Dec. 2011.
[6] S. Lai and B. Ravindran, "On distributed time-dependent shortest paths over duty-cycled wireless sensor networks," in Proc. IEEE INFOCOM, 2010, pp. 1–9.
[7] S. Xiong, J. Li, M. Li, J. Wang, and Y. Liu, "Multiple task scheduling for low-duty-cycled wireless sensor networks," in Proc. IEEE INFOCOM, 2011, pp. 1323–1331.
[8] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in Proc. 2nd Int. Conf. Embedded Netw. SenSys, 2004, pp. 39–49.