Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 31 (2008) 999–1011
computer communications
www.elsevier.com/locate/comcom

Quick convergecast in ZigBee beacon-enabled tree-based wireless sensor networks

Meng-Shiuan Pan *, Yu-Chee Tseng

Department of Computer Science, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsin-Chu 30010, Taiwan

Available online 25 December 2007

Abstract

Convergecast is a fundamental operation in wireless sensor networks. Existing convergecast solutions have focused on reducing latency and energy consumption. However, a good design should be compliant to standards, in addition to considering these factors. Based on this observation, this paper defines a *minimum delay beacon scheduling problem* for quick convergecast in ZigBee tree-based wireless sensor networks and proves that this problem is NP-complete. Our formulation is compliant with the low-power design of IEEE 802.15.4. We then propose optimal solutions for special cases and heuristic algorithms for general cases. Simulation results show that the proposed algorithms can indeed achieve quick convergecast.

Keywords: Convergecast; IEEE 802.15.4; Scheduling; Wireless sensor network; ZigBee

1. Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks* (WSNs) possible. A WSN consists of many inexpensive wireless sensors capable of collecting, storing, processing environmental information, and communicating with neighboring nodes. Applications of WSNs include wildlife monitoring [3,4], object tracking [16,18], and dynamic path finding [15,19].

Recently, several WSN platforms have been developed, such as MICA [6] and Dust Network [2]. For interoperability among different systems, standards such as ZigBee [24] have been developed. In the ZigBee protocol stack, physical and MAC layer protocols are adopted from the IEEE 802.15.4 standard [13]. ZigBee solves interoperability issues from the physical layer to the application layer.

ZigBee supports three kinds of networks, namely *star*, *tree*, and *mesh* networks. A *ZigBee coordinator* is responsible for initializing, maintaining, and controlling the network. A star network has a coordinator with devices directly connecting to the coordinator. For tree and mesh networks, devices can communicate with each other in a multihop fashion. The network is formed by one ZigBee coordinator and multiple *ZigBee routers*. A device can join a network as an *end devices* by the associating with the coordinator or a router. In a tree network, the coordinator and routers can announce beacons. However, in a mesh network, regular beacons are not allowed. Beacons are an important mechanism to support power management. Therefore, the tree topology is preferred, especially when energy saving is a desirable feature. To support ZigBee beacon-enabled tree networks, the IEEE 802.15 WPAN Task Group 4 further defines a revision of the IEEE 802.15.4 [14] specification in 2006. One of the major changes is structure of superframes to support power management. On the contrary, to our understanding, power management is still impossible for mesh-based ZigBee networks in the current specification. Therefore, we will focus on tree-based, beacon-enabled ZigBee networks in this work.

Considering that data gathering is a major application of WSNs, *convergecast* has been investigated in several

* Corresponding author. Tel.: +886 933243597.
E-mail addresses: mspan@cs.nctu.edu.tw (M.-S. Pan), yctseng@cs.nctu.edu.tw (Y.-C. Tseng).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.12.015

works [8,9,11,17,20,23]. With the goals of low latency and low energy consumption, Ref. [20] shows how to connect sensors as a balanced reporting tree and how to assign CDMA codes to sensors to diminish interference among sensors, thus achieving energy efficiency. The work [23] aims to minimize the overall energy consumption under the constraint that sensed data should be reported within specified time. Dynamic programming algorithms are proposed by assuming that sensors can receive multiple packets at the same time. As can be seen, both [20] and [23] are based on quite strong assumptions on communication capability of sensor nodes and they do not fit into the ZigBee specification. In [17], the authors propose an energy-efficient and low-latency MAC, called *DMAC*. Sensors are connected by a tree and stay in sleep state for most of the time. When sensors change to active state, they are first set to the receive mode and then to the transmit mode. *DMAC* achieves low-latency by staggering wake-up schedules of sensors at the time instant when their children switch to the transmit mode. Similar to [17], Ref. [11] arranges wake-up schedule of sensors by taking traffic loads into account. Each parent periodically broadcasts an advertisement containing a set of empty slots. Children nodes request empty slots according to their demands. In [9], the authors propose a distributed convergecast scheduling algorithm. The basic concept is to connect nodes by a spanning tree. Then the algorithm reduces the tree to multiple lines. For each line, the algorithm schedules nodes' transmission times in a bottom-up manner. Ref. [8] presents a centralized solution to convergecast. The algorithm divides nodes into many segments such that the transmission of a node in a segment does not cause interference to other transmissions in the same segment. The aim is to increase the degree of parallel transmissions to decrease latencies. Although these results [8,9,11,17] are designed for quick convergecast, the solutions are not compliant to the ZigBee standard for the following two reasons. Firstly, in these works, nodes' wake/sleep times are dynamically changed according to their schedules. However, in a ZigBee beacon-enabled tree network, nodes' wake/sleep times must be fixed in the way that each router wakes up twice in each cycle to receive its children's packets and to transmit packets to its parent, respectively. The coordinator (resp., an end device) wakes up once to receive its children's packets (resp., to transmit packets to its parent). Secondly, the scheduling of [8,9,11,17] is transmission-based, while ours are receiving-based. The implication is that the former may cause a router to be active multiple times per cycle. This is incompatible with the ZigBee specification.

This paper aims at designing quick convergecast solutions for ZigBee tree-based, beacon-enabled WSNs. This work is motivated by the following observations. First, we see that most related works are not compliant to the ZigBee standard. Second, we believe that tree-based topology is more suitable if power management is a main concern in WSNs. The network scenario is shown in Fig. 1. The network contains one *sink* (ZigBee coordinator), some ZigBee routers, and some ZigBee end devices. Each ZigBee router is responsible for collecting sensed data from end devices associated with it and relaying incoming data to the sink. According to specifications, a ZigBee router can announce a beacon to start a superframe. Each superframe consists of an *active portion* followed by an *inactive portion*. On receiving its parent router's beacon, an end device has to wake up for an active portion to sense the environment and communicate with its coordinator. However, to avoid collision with its neighbors, a router should shift its active portion by a certain amount. Fig. 1 shows a possible allocation of active portions for routers A, B, C, and D. The collected sensory data of A in the k-th superframe can be sent to C in the same superframe. However, because the active portion of B in the k-th superframe appears after that of C, the collected data of B in the k-th superframe can only be relayed to C in the $(k+1)$-th superframe. The report delay from B to C is almost the length of one superframe. The delay can be eliminated if the active portion of B in the k-th superframe appears before that of C. The delay is not negligible because of the low duty cycle design of IEEE 802.15.4. For example, in 2.4 GHz PHY, with 1.56% duty cycle, a superframe can be as long as 251.658 s (with an active portion of 3.93 s). Clearly, for large-scale WSNs, the convergecast latency could be significant if the problem is not carefully addressed. The quick convergecast problem is to schedule the beacons of routers to minimize the convergecast latency. We prove that this problem is NP-complete by reducing the 3-CNF-SAT problem to it. We show two special cases of this problem where optimal solutions can be found in polynomial time and propose two heuristic algorithms for general cases. To the best of our knowledge, this is the first result that provides convergecast solutions in ZigBee beacon-enabled tree networks.

The rest of this paper is organized as follows. Section 2 briefly introduces IEEE 802.15.4 and ZigBee. The quick convergecast problem is formally defined in Section 3. Section 4 presents our scheduling solutions. Simulation results are given in Section 5. Finally, Section 6 concludes this paper.

2. Overview of IEEE 802.15.4 and ZigBee standards

IEEE 802.15.4 [13] specifies the physical and data link protocols for *low-rate wireless personal area networks* (LR-WPAN). In the physical layer, there are three frequency bands with 27 radio channels. Channel 0 ranges from 868.0 to 868.6 MHz, which provides a data rate of 20 kbps. Channels 1 to 10 work from 902.0 to 928.0 MHz and each channel provides a data rate of 40 kbps. Channels 11 to 26 are located from 2.4 to 2.4835 GHz, each with a data rate of 250 kbps.

IEEE 802.15.4 devices are expected to have limited power, but need to operate for a longer period of time. Therefore, energy conservation is a critical issue. Devices are classified as *full function devices* (FFDs) and *reduced*

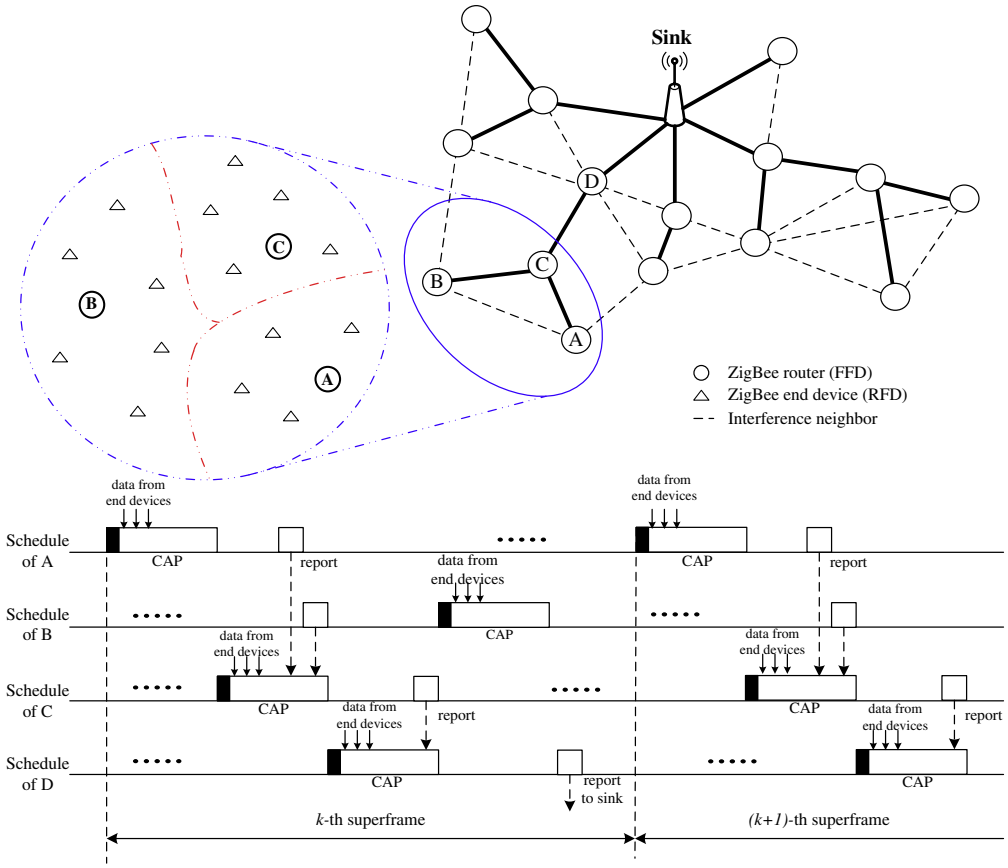

Fig. 1. An example of convergecast in a ZigBee tree-based network.

function devices (RFDs). IEEE 802.15.4 supports star and peer-to-peer topologies. In each PAN, one device is designated as the *coordinator*, which is responsible for maintaining the network. A FFD has the capability of serving as a coordinator or associating with an existing coordinator/router and becoming a router. A RFD can only associate with a coordinator/router and can not have children.

The ZigBee coordinator defines the superframe structure of a ZigBee network. As shown in Fig. 2(a), the structure of superframes is controlled by two parameters: *beacon order* (BO) and *superframe order* (SO), which decide the lengths of a superframe and its active potion, respectively. For a beacon-enabled network, the setting of BO and SO should satisfy the relationship $0 \leqslant SO \leqslant BO \leqslant 14$. (A non-beacon-enabled network should set $BO = SO = 15$ to indicate that superframes do not exist.) Each active portion consists of 16 equal-length slots, which can be further partitioned into a *contention access period* (CAP) and a *contention free period* (CFP). The CAP may contain the first i slots, and the CFP contains the rest of the $16 - i$ slots, where $1 \leqslant i \leqslant 16$. Slotted CSMA/CA is used in CAP. FFDs which require fixed transmission rates can ask for *guarantee time slots* (GTSs) from the coordinator. A CFP can support multiple GTSs, and each GTS may contain multiple slots. Note that only the coordinator can allocate GTSs. After the active portion, devices can go to sleep to save energy.

In a beacon-enabled star network, a device only needs to be active for $2^{-(BO-SO)}$ portion of the time. Changing the value of $(BO - SO)$ allows us to adjust the on-duty time of devices. However, for a beacon-enabled tree network, routers have to choose different times to start their active portions to avoid collision. Once the value of $(BO - SO)$ is decided, each router can choose from 2^{BO-SO} slots as its active portion. In the revised version of IEEE 802.15.4 [14], a router can select one active portion as its *outgoing superframe*, and based on the active portion selected by its parent, the active portion is called its *incoming superframe* (as shown in Fig. 2(b)). In an outgoing/incoming superframe, a router is expected to transmit/receive a beacon to/from its child routers/parent router. When choosing a slot, neighboring routers' active portions (i.e., outgoing superframes) should be shifted away from each other to avoid interference. This work is motivated by the observation that the specification does not clearly define how to choose the locations of routers' active portions such that the convergecast latency can be reduced. In our work, we consider two kinds of interference between routers. Two routers have *direct interference* if they can hear each others' beacons. Two routers have *indirect interference* if they have at least one common neighbor. Both interferences should be avoided when choosing routers' active portions. Table 1 lists possible choices of $(BO - SO)$ combinations.

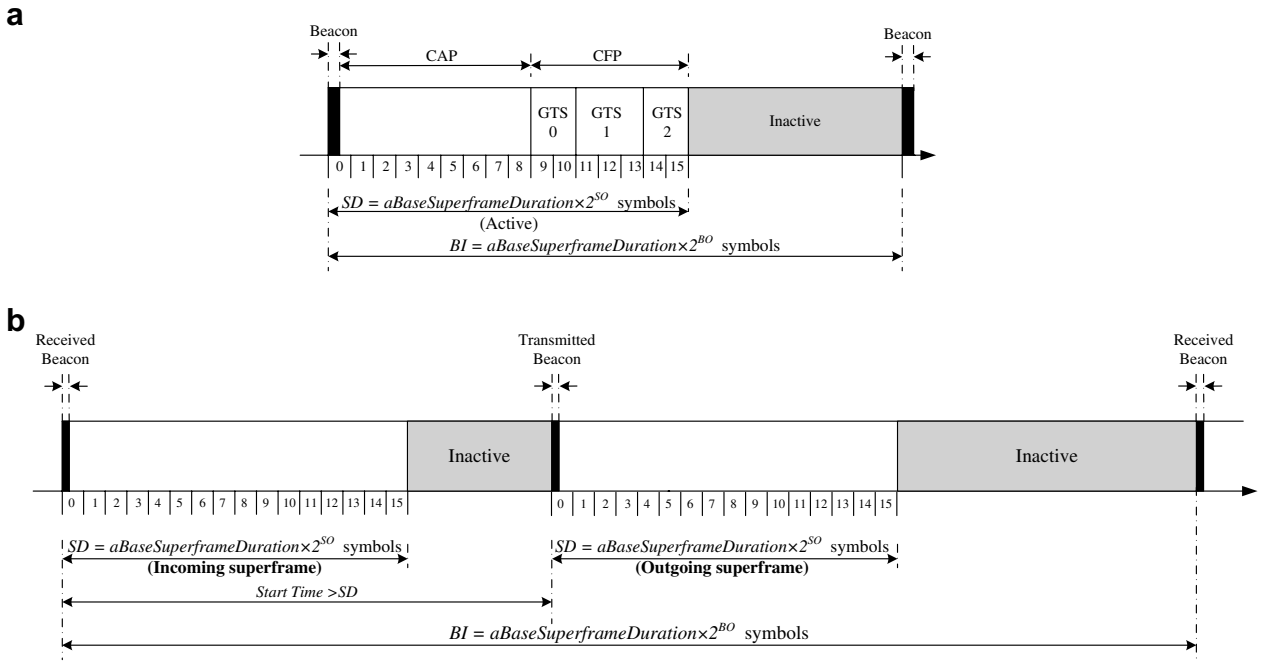

Fig. 2. IEEE 802.15.4 superframe structure.

Table 1
Relationship of BO − SO, duty cycle, and the number of active portions in a superframe

BO − SO	0	1	2	3	4	5	6	7	8	⩾9
Duty cycle (%)	100	50	25	12.5	6.25	3.13	1.56	0.78	0.39	⩽0.195
Number of active portions (slots)	1	2	4	8	16	32	64	128	256	⩾512

3. The minimum delay beacon scheduling (MDBS) problem

This section formally defines the convergecast problem in ZigBee networks. Given a ZigBee network, we model it by a graph $G = (V, E)$, where V contains all routers and the coordinator and E contains all symmetric communication links between nodes in V. The coordinator also serves as the sink of the network. End devices can only associate with routers, but are not included in V. From G, we can construct an *interference graph* $G_I = (V, E_I)$, where edge $(i, j) \in E_I$ if there are direct/indirect interferences between i and j. There is a duty cycle requirement α for this network. From α and Table 1, we can determine the most appropriate value of BO − SO. We denote by $k = 2^{BO-SO}$ the number of active portions (or slots) per beacon interval.

The beacon scheduling problem is to find a slot assignment $s(i)$ for each router $i \in V$, where $s(i)$ is an integer and $s(i) \in [0, k-1]$, such that router i's active portion is in slot $s(i)$ and $s(i) \neq s(j)$ if $(i, j) \in E_I$. Here, the slot assignment means the position of the outgoing superframe of each router (the position of the incoming superframe, as clarified earlier, is determined by the parent of the router). Motivated by Brook's theorem [21], which proves that n colors are sufficient to color any graph with a maximum degree of n, we would assume that $k \geqslant D_I$, where D_I is the maximum degree of G_I.

Given a slot assignment for G, the report latency from node i to node j, where $(i, j) \in E$, is the number of slots, denoted by d_{ij}, that node i has to wait to relay its collected sensory data to node j, i.e.,

$$d_{ij} = (s(j) - s(i)) \bmod k. \quad (1)$$

Note that the report latency from node i to node j (d_{ij}) may not by equal to the report latency from node j to node i (d_{ji}). Therefore, we can convert G into a weighted directed graph $G_D = (V, E_D)$ such that each $(i, j) \in E$ is translated into two directed edges (i, j) and (j, i) such that $w((i, j)) = d_{ij}$ and $w((j, i)) = d_{ji}$. The report latency for each $i \in V$ to the sink is the sum of report latencies of the links on the shortest path from i to the sink in G_D. The latency of the convergecast, denoted as $L(G)$, is the maximum of all nodes' report latencies.

Definition 1. Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, and k available slots, the *minimum delay beacon scheduling* (MDBS) *problem* is to find an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G)$ is minimized.

To prove that the MDBS problem is NP-complete, we define a decision problem as follows.

Definition 2. Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, k available slots, and a delay constraint d,

the *bounded delay beacon scheduling* (BDBS) *problem* is to decide if there exists an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G) \leqslant d$.

Theorem 1. *The BDBS problem is NP-complete.*

Proof. First, given slot assignments for nodes in V, we can find the report latency of each $i \in V$ by running a shortest path algorithm on G_D. We can then check if $L(G) \leqslant d$. Clearly, this takes polynomial time.

We then prove that the BDBS problem is NP-hard by reducing the 3 conjunctive normal form satisfiability (3-CNF-SAT) problem to a special case of the BDBS problem in polynomial time. Given any 3-CNF formula C, we will construct the corresponding G and G_I. Then we show that C is satisfiable *if and only if* there is a slot assignment for each $i \in V$ using no more than $k = 3$ slots such that $L(G) \leqslant 4$ slots.

Let $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where clause $C_j = x_{j,1} \vee x_{j,2} \vee x_{j,3}$, $1 \leqslant j \leqslant m$, $x_{j,i} \in \{X_1, X_2, \ldots, X_n\}$, and $X_i \in \{x_i, \bar{x}_i\}$, where x_i is a binary variable, $1 \leqslant i \leqslant n$. We first construct G from C as follows:

1. For each clause C_j, $j = 1, 2, \ldots, m$, add a vertex C_j in G.
2. For each literal X_i, $i = 1, 2, \ldots, n$, add four vertices x_{i1}, x_{i2}, $\bar{x}_{i1}$, and $\bar{x}_{i2}$ in G.
3. Add a vertex t as the sink of G.
4. Add edges (t, x_{i2}) and $(t, \bar{x}_{i2})$ to G, for $i = 1, 2, \ldots, n$.
5. Add edges (x_{i1}, x_{i2}) and $(\bar{x}_{i1}, \bar{x}_{i2})$ to G, for $i = 1, 2, \ldots, n$.
6. For each $i = 1, 2, \ldots, n$ and each $j = 1, 2, \ldots, m$, add an edge (C_j, x_{i1}) (resp., $(C_j, \bar{x}_{i1})$) to G if x_i (resp., $\bar{x}_i$) appears in C_j.

Then we construct G_I as follows.

1. Add all vertices and edges in G into G_I.
2. Add edges $(x_{i1}, \bar{x}_{i1})$ and $(x_{i2}, \bar{x}_{i2})$ to G_I, for $i = 1, 2, \ldots, n$.
3. Add edges (C_j, x_{i2}) and $(C_j, \bar{x}_{i2})$ to G_I, for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

Then we build a one-to-one mapping from each truth assignment of C to a slot assignment of G. We establish the following mapping:

1. Set $s(t) = 0$.
2. Set $s(C_j) = 0$, $j = 1, 2, \ldots, m$.
3. Set $s(x_{i1}) = 1$ and $s(\bar{x}_{i2}) = 1$, $i = 1, 2, \ldots, n$, if x_i is true; otherwise, set $s(x_{i1}) = 2$ and $s(\bar{x}_{i2}) = 2$.
4. Set $s(x_{i2}) = 1$ and $s(\bar{x}_{i1}) = 1$, $i = 1, 2, \ldots, n$, if $\bar{x}_i$ is true; otherwise, set $s(x_{i2}) = 2$ and $s(\bar{x}_{i1}) = 2$.

The above reduction can be computed in polynomial time. By the above reduction, vertices x_{i1} or $\bar{x}_{i1}$, $i = 1, 2, \ldots, n$, that are assigned to slot 1 (resp., slot 2) will have a report latency of 2 (resp., 4) and vertices x_{i2} or $\bar{x}_{i2}$, $i = 1, 2, \ldots, n$, that are assigned to slot 1 (resp., slot 2) will have a report latency of 2 (resp., 1). Hence, for those vertices x_{i1}, $\bar{x}_{i1}$, x_{i2}, and $\bar{x}_{i2}$, $i = 1, 2, \ldots, n$, the longest report latency will be 4.

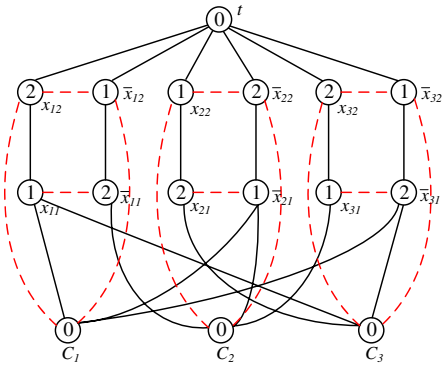

Fig. 3. An example of reduction from the 3-CNF-SAT to the BDBS problem.

To prove the *if* part, we need to show that if C is satisfiable, there is a slot assignment such that $k = 3$ and $L(G) \leqslant 4$. Since C satisfiable, there must exist an assignment such that each clause C_j, $j = 1, 2, \ldots, m$, is true. If a clause C_j is true, at least one variable in C_j is true. According to the reduction, C_j can always find an edge (C_j, x_{i1}) or $(C_j, \bar{x}_{i1})$ with $w((C_j, x_{i1})) = 1$ or $w((C_j, \bar{x}_{i1})) = 1$, where $i = 1, 2, \ldots, n$. Thus, when C is satisfiable, the reporting latency for each clause is 3. This achieves $L(G) = 4$.

For the *only if* part, if each vertex C_j, $j = 1, 2, \ldots, m$, can find at least an edge with weight 1 to one of x_{i1} and $\bar{x}_{i1}$, for $i = 1, 2, \ldots, n$, to achieve a report latency of 3, it must be that each clause has at least one variable to be true. So formula C is satisfiable. Otherwise, the report latency of C_j, $j = 1, 2, \ldots, m$, will be 6. □

For example, given $C = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$, Fig. 3 shows the corresponding G. The truth assignment $(x_1, x_2, x_3) = (T, F, T)$ makes C satisfiable. According to the reduction and the mapping in the above proof, we can obtain the network G and its slot assignment as shown in Fig. 3 such that $L(G) = 4$.

4. Algorithms for the MDBS problem

4.1. Optimal solutions for special cases

Optimal solutions can be found for the MDBS problem in polynomial time for regular linear networks and regular ring networks, as illustrated in Fig. 4. In such networks, each vertex is connected to one or two adjacent vertices and has an interference relation with each neighbor within h hops from it, where $h \geqslant 2$. In a regular linear network, we assume that the sink t is at one end of the network. Clearly, the maximum degree of G_I is $2h$. We will show that an optimal solution can be found if the number of slots $k \geqslant h + 1$. The slot assignment can be done in a bottom-up manner. The bottom node is assigned to slot 0. Then,

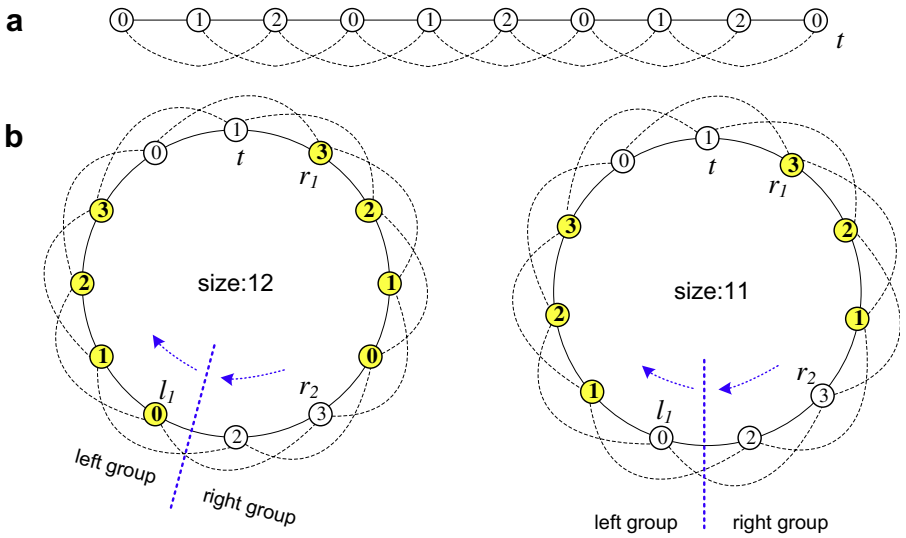

Fig. 4. Examples of optimal slot assignments for regular linear and ring networks ($h = 2$). Dotted lines mean interference relations.

for each vertex v, $s(v) = (k\prime + 1) \bmod k$, where $k\prime$ is the slot assigned to v's child.

Theorem 2. *For a regular linear network, if $k \geqslant h + 1$, the above slot assignment achieves a report latency of $|V| - 1$, which is optimal.*

Proof. Clearly, the slot assignment is interference-free. Also the report latency of $|V| - 1$ is clearly the lower bound. □

For a regular ring network, we first partition vertices excluding t into left and right groups as illustrated in Fig. 4(b) such that the left group consists of the sink node t and $\lfloor \frac{|V|-1}{2} \rfloor$ other nodes counting counter clockwise from t, and the right group consists of those $\lceil \frac{|V|-1}{2} \rceil$ nodes counting clockwise from t. Now we consider the ring as a spanning tree with t as the root and left and right groups as two linear paths. Assuming that $\lfloor \frac{|V|-1}{2} \rfloor \geqslant 2h$ and $k \geqslant 2h$, the slot assignment works as follows:

1. The bottom node in the left group is assigned to slot 0.
2. All other nodes in the left group are assigned with slots in a bottom-up manner. For each node i in the left group, we let $s(i) = (j + 1) \bmod k$, where j is the slot of i's child.
3. Nodes in the right group are assigned with slots in a top-down manner. For each node i in the right group, we let $s(i) = (j - c) \bmod k$, where j is the slot assigned to i's parent and c is the smallest constant ($1 \leqslant c \leqslant k$) that ensures that $s(i)$ is not used by any of its interference neighbors that have been assigned with slots.

It is not hard to prove the slot assignment is interference-free because nodes receives slots sequentially and we have avoided using the same slots among interfering neighbors. Although this is a greedy approach, we show that c is equal to 1 in step 3 in most of the cases except when two special nodes are visited. This gives an asymptotically optimal algorithm, as proved in the following theorem.

Theorem 3. *For a regular ring network, assuming that $k \geqslant 2h$ and $\lfloor \frac{|V|-1}{2} \rfloor \geqslant 2h$, the above slot assignment achieves a report latency $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$, which is optimal within a factor of* 1.5.

Proof. We first identify three nodes on the ring (refer to Fig. 4(b)):

- l_1: the bottom node in the left group.
- r_1: the first node in the right group.
- r_2: the node that is h hops from l_1 counting counter clockwise.

The report latency of each node can be analyzed as follows. The parent of node x is denoted by $par(x)$.

A1. For each node i in the left group except the sink t, the latency from i to $par(i)$ is 1.
A2. The latency from r_1 to t is h.
A3. For each node i next to r_1 in the right group but before r_2 (counting clockwise), the latency from i to $par(i)$ is 1.
A4. The latency from r_2 to $par(r_2)$ is 1 if the ring size is even; otherwise, the latency is 2.
A5. For each node i in the right group that is a descendant of r_2, the report latency from i to $par(i)$ is 1.

It is not hard to prove that A1, A2, and A3 are true. To see A4 and A5, we make the following observations. The function $par^i(x)$ is to apply i times the $par()$ function on node x. Note that $par^0(x)$ means x itself.

O1. When the ring size is even, the equality $s(par^{i-1}(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, \ldots, \lfloor\frac{|V|-1}{2}\rfloor - h - 1$. More specifically, this means that (i) l_1 and $par(r_2)$ will receive the same slot, (ii) $par(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. This can be proved by induction by showing that the i-th descendant of t in the right group will be assigned the same slot as the $(h+i-1)$-th descendant of t in the left group (the induction can go in a top-down manner). This property implies that when assigning a slot to r_2 in step 3, $c = 1$ in case that the ring size is even. Further, r_2 and its descendants will be sequentially assigned to slots $k-1$, $k-2$, …, $k-h$, which implies that $c = 1$ when doing the assignments in step 3. So properties A4 and A5 hold for the case of an even ring.

O2. When the ring size is odd, the equality $s(par^i(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, \ldots, \lfloor\frac{|V|-1}{2}\rfloor - h$. This means that (i) $par(l_1)$ and $par(r_2)$ will receive the same slot, and (ii) $par^2(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. Again, this can be proved by induction as in O1. This property implies that $c = 2$ when assigning a slot to r_2 in step 3, and $c = 1$ when assigning slots to descendants of r_2. So properties A4 and A5 hold for the case of an odd ring.

The equality of slot assignments pointed out in O1 and O2 is illustrated in Fig. 4(b) by those numbers in gray nodes. In summary, the report latency of the left group is $\lfloor\frac{|V|-1}{2}\rfloor$. When the ring size is even, the report latency of the right group is the number of nodes in this group, $\frac{|V|}{2}$, plus the extra latency $h-1$ incurred at r_1. So $L(G) = \frac{|V|}{2} + h - 1 = \lfloor\frac{|V|-1}{2}\rfloor + h$. When the ring size is odd, the report latency of right group is the number of nodes in this group, $\frac{|V|-1}{2}$, plus the extra latency $h-1$ incurred at r_1 and the extra latency 1 incurred at r_2. So $L(G) = \lfloor\frac{|V|-1}{2}\rfloor + h$.

A lower bound on the report latency of this problem is the maximum number of nodes in each group excluding t. Applying $\lfloor\frac{|V|-1}{2}\rfloor$ as a lower bound and using the fact that $\lfloor\frac{|V|-1}{2}\rfloor \geqslant 2h$, $L(G)$ will be smaller than $1.5 \times \lfloor\frac{|V|-1}{2}\rfloor$, which implies the algorithm is optimal within a factor of 1.5. Note that the condition $\lfloor\frac{|V|-1}{2}\rfloor \geqslant 2h$ is to guarantee that t will not locate within h hops from r_2. Otherwise, the observation O2 will not hold. □

4.2. A centralized tree-based assignment scheme

Given $G = (V, E)$, $G_I = (V, E_I)$, and k, we propose a centralized slot assignment heuristic algorithm. Our algorithm is composed of the following three phases:

phase 1. From G, we first construct a BFS tree T rooted at sink t.

phase 2. We traverse vertices of T in a bottom-up manner. For these vertices in depth d, we first sort them according to their degrees in G_I in a descending order. Then we sequentially traverse these vertices in that order. For each vertex v in depth d visited, we compute a temporary slot number $t(v)$ for v as follows.

1. If v is a leaf node, we set $t(v)$ to the minimal non-negative integer l such that for each vertex u that has been visited and $(u, v) \in E_I$, $(t(u) \bmod k) \neq l$.
2. If v is an in-tree node, let m be the maximum of the numbers that have been assigned to v's children, i.e., $m = \max\{t(child(v))\}$, where $child(v)$ is the set of v's children. We then set $t(v)$ to the minimal non-negative integer $l > m$ such that for each vertex u that has been visited and $(u, v) \in E_I$, $(t(u) \bmod k) \neq (l \bmod k)$.

After every vertex v is visited, we make the assignment $s(v) = t(v) \bmod k$.

phase 3. In this phase, vertices are traversed sequentially from t in a top-down manner. When each vertex v is visited, we try to greedily find a new slot l such that $(s(par(v)) - l) \bmod k < (s(par(v)) - s(v)) \bmod k$, such that $l \neq s(u)$ for each $(u, v) \in E_I$, if possible. Then we reassign $s(v) = l$.

Note that in phase 2, a node with a higher degree means that it has more interference neighbors, implying that it has less slots to use. Therefore, it has to be assigned to a slot earlier. Also note that, the number $t(v)$ is not a modulus number. However, in step 2 of phase 2, we did check that if $t(v)$ is converted to a slot number, no interference will occur. Intuitively, this is a temporary slot assignment that will incur the least latency to v's children. At the end, $t(v)$ is converted to a slot assignment $s(v)$. Phase 3 is a greedy approach to further reduce the report latency of routers. For example, Fig. 5(a) shows the slot assignment after phase 2. Fig. 5(b) indicates that B, C, and D can find another slots and their report latencies are decreased. This phase can reduce $L(G)$ in some cases.

The computational complexity of this algorithm is analyzed below. In phase 1, the complexity of constructing a BFS tree is $O(|V| + |E|)$. In phase 2, the cost of sorting is at most $O(|V|^2)$ and the computational cost to compute $t(v)$ for each vertex v is bounded by $O(kD_I)$, where D_I is the degree of G_I. So the time complexity of phase 2 is $O(|V|^2 + kD_I|V|)$. Phase 3 performs a similar procedure as phase 2, so its time complexity is also $O(kD_I|V|)$. Overall, the time complexity is $O(|V|^2 + kD_I|V|)$.

4.3. A distributed assignment scheme

In this section, we propose a distributed slot assignment algorithm. Each node has to compute its direct as well as indirect interference neighbors in a distributed manner. To achieve this, we will refer to the *heterogeneity* approach

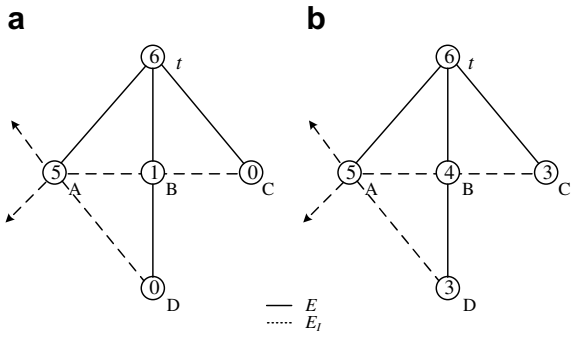

Fig. 5. (a) Slot assignment after phase 2. (b) Slot compacting by phase 3.

in [22], which adopts power control to achieve this goal. Assuming routers' default transmission range is r, interference neighbors must locate within range $2r$. From time-to-time, each router will boost its transmission power to double its default transmission range and send HELLO packets to its neighbor routers. Each HELLO packet further contains sender's (1) depth,[1] (2) the location of outgoing superframe (i.e., slot), and (3) number of interference neighbors. Note that all other packets are transmitted by the default power level. When booting up, each router will broadcast HELLO packets claiming that its depth and slot are *NULL*. After joining the network and choosing a slot, the HELLO packets will carry the node's depth and slot information. The algorithm is triggered by the sink t setting $s(t) = k - 1$ and then broadcasting its beacon. A router $v \neq t$ that receives a beacon will decide its slot as follows.

1. Node v sends an association request to the beacon sender.
2. If v fails to associate with the beacon sender, it stops the procedure and waits for other beacons.
3. If v successfully associates with a parent node $par(v)$, it computes the smallest positive integer l such that $(s(par(v)) - l) \bmod k \neq s(u)$ for all $(u, v) \in E_I$ and $s(u) \neq NULL$. Then v chooses $s(v) = (s(par(v)) - l) \bmod k$ as its slot.
4. Then, v broadcasts HELLOs including its slot assignment $s(v)$ for a time period t_{wait}. If it finds that $s(v) = s(u)$ for any $(u, v) \in E_I$, v has to change to a new slot if one of the following rules is satisfied and goes back to step 3.
   (a) Node u has more interference neighbors than v.
   (b) Node u and v have the same number of interference neighbors but the depth of u is lower than v, i.e., u is closer to the sink than v.
   (c) Node u and v have the same number of interference neighbors and they are at the same depth but the u's ID is smaller than v's.
5. After t_{wait}, v can finalize its slot selection and broadcast its beacons.

In this distributed algorithm, slots are assigned to routers, ideally, in a top-down manner. However, due to transmission latency, some routers at lower levels may find slots earlier than those at higher levels. Also note that the time t_{wait} is to avoid possible collision on slot assignments due to packet loss.

5. Simulation results

This section presents our simulation results. We first assume that the size of sensory data is negligible and that all routers generate reports at the same time, and compare the performances of different convergecast algorithms. Then we simulate more realistic scenarios where the size of sensory data is not negligible and routers need to generate reports periodically or passively driven by events randomly appearing in certain regions in the sensing field. More specifically, sensors generate reports according to certain application specifications. Devices all run ZigBee and IEEE 802.15.4 protocols to communicate with each other. Routers can aggregate child sensors' reports and report to their parents directly. Each router has a fix-size buffer. When a router's buffer overflows, this router will not accept further incoming frames. We also measure the *goodput* of the network, which is defined as the ratio of sensors' reports successfully received by the sink. Some parameters used in our simulation are listed in Table 2.

5.1. Comparison of different convergecast algorithms

We compare the proposed slot assignment algorithms against a random slot assignment (denoted by RAN) scheme and a greedy slot assignment (denoted by GDY) scheme. In RAN, the slot assignment starts from the sink and each router, after associating with a parent router, simply chooses any slot which has not been used by any of its interference neighbors. In GDY, routers are given a sequence number in a top-down manner. The sink sets its slot to $k - 1$. Then the slot assignment continues in sequence. For a node i, it will try to find a slot $s(i) = s(j) - l \bmod k$, where j is the predecessor of i and l is the smallest integer letting $s(i)$ is the slot which does not assign to any of i's interference neighbors. In the simulations, routers are randomly distributed in a circular region of a radius r and a sink is placed in the center. Our centralized tree-based scheme and distributed slot assignment scheme are denoted as CTB and DSA, respectively. We compare the report latency $L(G)$ (in terms of slots).

Fig. 6 shows some slot assignment results of CTB and DSA when $r = 35$ m and $k = 64$. Devices are randomly distributed. The transmission range of routers is set to 20 m. In this case, CTB performs better than DSA.

[1] The depth of a node is the length of the tree path from the root to the node. The root node is at depth zero.

Table 2
Simulation parameters

Parameter	Value
Length of a frame's header and tail	18 Bytes
Length of a sensor's report	16 Bytes
Beacon length	18 Bytes
Maximum length of a frame	127 Bytes
Bit rate	250 kbps
Symbol rate	62.5 k symbols/s
aBaseSuperframeDuration	960 symbols
aUnitBackoffPeriod	20 symbols
aCCATime	8 symbols
MacMinBE	3
aMaxBE	5
MacMaxCSMABackoffs	4
Maximum number of retransmissions	3

Next, we observe the impact of different r, C_R (number of routers), and T_R (transmission distance). Fig. 7(a) shows the impact of r when $k = 64$, $T_R = 25$ m, and $C_R = 3 \times (r/10)^2$. CTB performs the best. DSA performs slightly worse than CTB, but still significantly outperforms RAN and GDY. It can be seen that RAN and GRY could result in very long convergecast latency. Both CTB and DSA are quite insensitive to the network size. But this is not the case for RAN and GDY. Fig. 7(b) shows the impact of T_R when $C_R = 300$, $r = 100$ m, and $k = 64$. Since a larger transmission range implies higher interference among routers, the report latencies of CTB and DSA will increase linearly as T_R increases. The report latency of RAN also increases when $T_R = 17$–21 m because of the increased interference. After $T_R \geqslant 22$ m, the latency of RAN decreases because that the network diameter is reduced. Basically, GDY behaves the same as CTB and DSA. But when the transmission range is larger, the report latency slightly becomes small.

Fig. 7(c) shows the impact of C_R when $r = 100$ m, $T_R = 20$ m, and $k = 128$. As a larger C_R means a higher network density and thus more interference, the report latencies of CTB and DSA increase as C_R increases. Since the network diameter is bounded, the report latency of RAN is also bounded. GDY is sensitive to the number of routers when there are less routers. This is because that each router can own a slot and the report latency increases proportionally to the number of routers. With $r = 100$ m, $C_R = 300$, and $T_R = 20$ m, Fig. 7(d) shows the impact of routers' duty cycle. Note that a lower duty cycle means a larger number of available slots. Interestingly, we see that the report latencies of CTB, DSA, and GDY are independent of the number of slots. Contrarily, with a random assignment, RAN even incurs a higher report latency as there are more freedom in slot selection.

5.2. Periodical reporting scenarios

Next, we assume that sensors are instructed to report their data in a periodically manner. We set $r = 100$ m, $T_R = 20$ m, and $C_R = 300$ with 6000 randomly placed sensors associated to these routers, and we further restrict a router can accept at most 30 sensors. BO − SO is fixed to six, so $k = 2^{BO-SO} = 64$. Since the earlier simulations show that CTB and DSA perform quite close, we will use only CTB to assign routers' slots. Sensors are required to generate a report every 251.66 s (the length of one beacon interval when BO = 14). We set the buffer size of each router is 10 KB.[2] We allocate two mini-slots for each child router of the sink as the GTS slot.[3]

Since (BO − SO) is fixed, a small BO implies a smaller slot size (and thus a smaller unit size of $L(G)$). So, a smaller slot size seemingly implies higher contention among sensors if they all intend to report to their parents simultaneously. In fact, a smaller BO does not hurt the overall reporting times of sensors if we can properly divide sensors into groups. For example, in Fig. 8, when BO = 14, all sensors of a router can report in every superframe. When BO = 13, if we divide sensors into two groups, then they can report alternately in odd and even superframes. Similarly, when BO = 12, four groups of sensors can report alternately. Since the length of superframes are reduced proportionally, the report intervals of sensors actually remain the same in these cases. In the following experiments, we groups sensors according to their parents' IDs. A sensor belongs to group m if the modulus of its parent's ID is m.

Fig. 9 shows the theoretical and actual report latencies under different BOs. Note that a report may be delayed due to buffer constraint. As can be seen, the actual latency does not always favor a smaller BO. Our results show that BO = 10–12 performs better. Fig. 9(b) shows the goodput of sensory reports, channel utilization at the sink, and the number of dropped frames at the sink. When BO = 14, although there is no frames being dropped at the sink, the goodput is still low. This is because a lot of collisions happen inside the network, causing many sensory reports being dropped at intermediate levels (a frame is dropped after exceeding its retransmission limit). Fig. 10 shows a log of the numbers of frames received by a sink's child router when BO = 14. We can see that more than half of the active portion is wasted. Overall, BO = 10 produces the best goodput and a shorter report latency.

Some previous works can be also integrated in this periodical reporting scenario, such as the adaptive GTS allocation mechanism in [12] and the aggregation algorithms for WSNs in [7,10]. Fig. 11 shows an experiment that routers can compress reports from sensors with a rate cr when BO = 10. If a router receives n reports and each report's size is 16 Bytes (as in Table 2), it can compress the size to $16 \times n \times (1 - cr)$. The report latencies decrease when the cr becomes larger. By compressing the report data,

[2] Currently, there are some platforms which are equipped with larger RAMs. For example, Jennic JN5121 [5] has a 96 KB RAM and CC2420DBK [1] has a 32 KB RAM.

[3] There are sixteen mini-slots per active portion (slot).

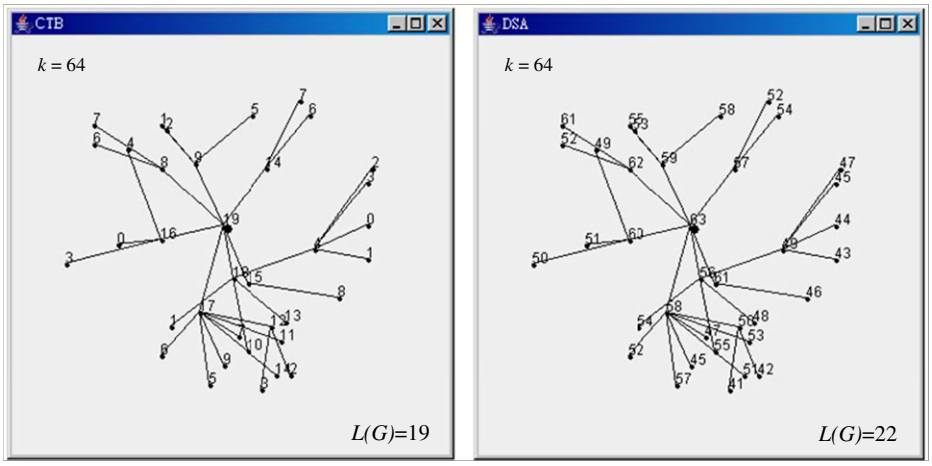

Fig. 6. Slot assignment examples by CTB and DSA.

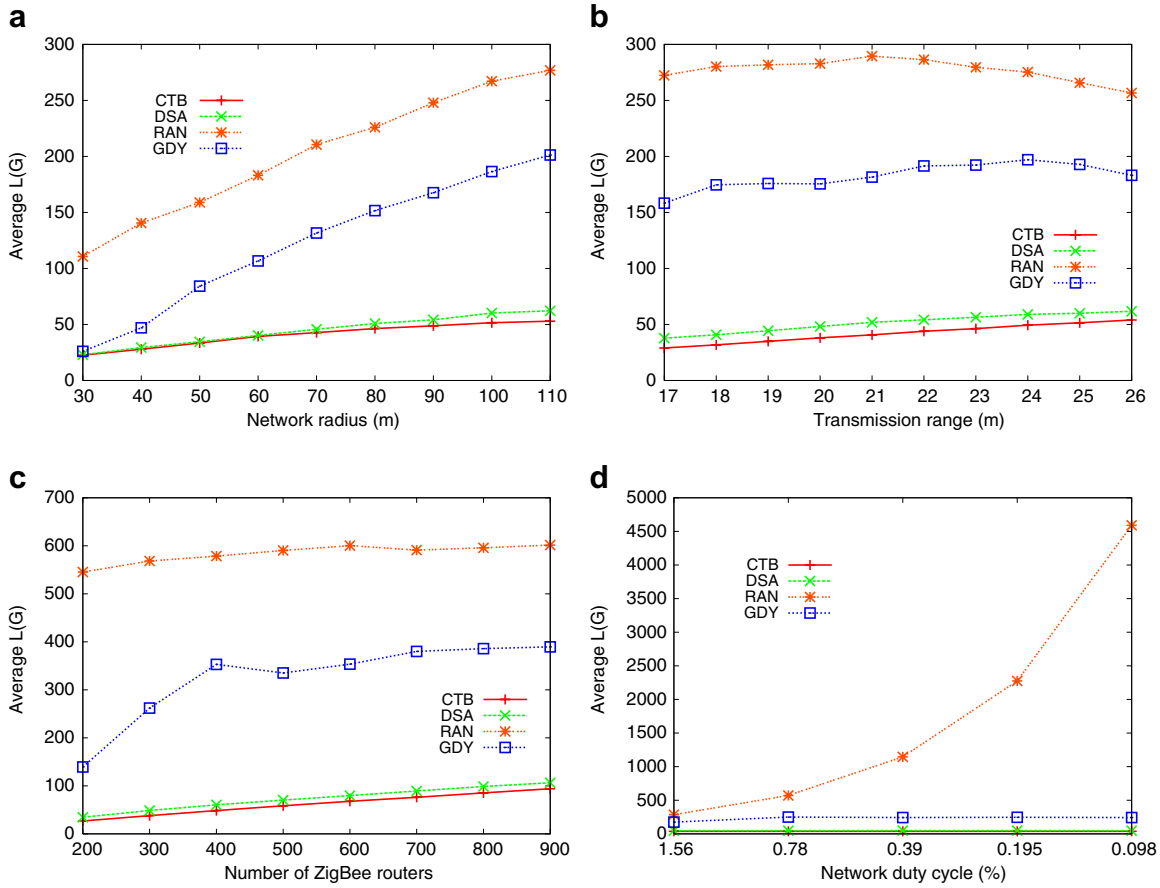

Fig. 7. Comparison of report latencies under different configurations.

the goodput can up to 98% and the report can arrive to the sink more quickly.

5.3. Event-driven reporting scenarios

In the following, we assume that sensors' reporting activities are triggered by events occurred at random locations in the network with a rate λ. The sensing range of each sensors is 3 m and each event is a disk of a radius of 5 m. A sensor can detect an event if its sensing range overlaps with the disk of that event. Each router has an 1 KB buffer. When a sensor detects an event, it only tries to report that event once. All other settings are the same as those in Section 5.2.

Fig. 12 shows the simulation results when $\lambda = 1/5$s, 1/15s, and 1/30s. From Fig. 12(a), we can observe that when

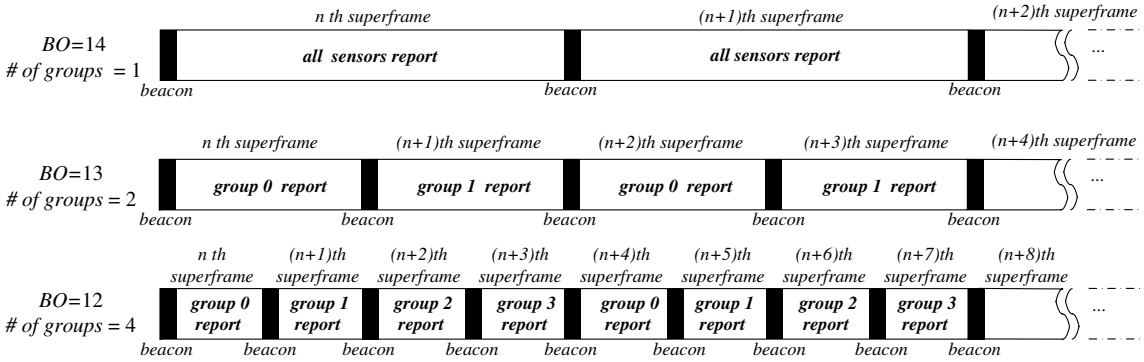

Fig. 8. An example of report scheduling under different values of BO.

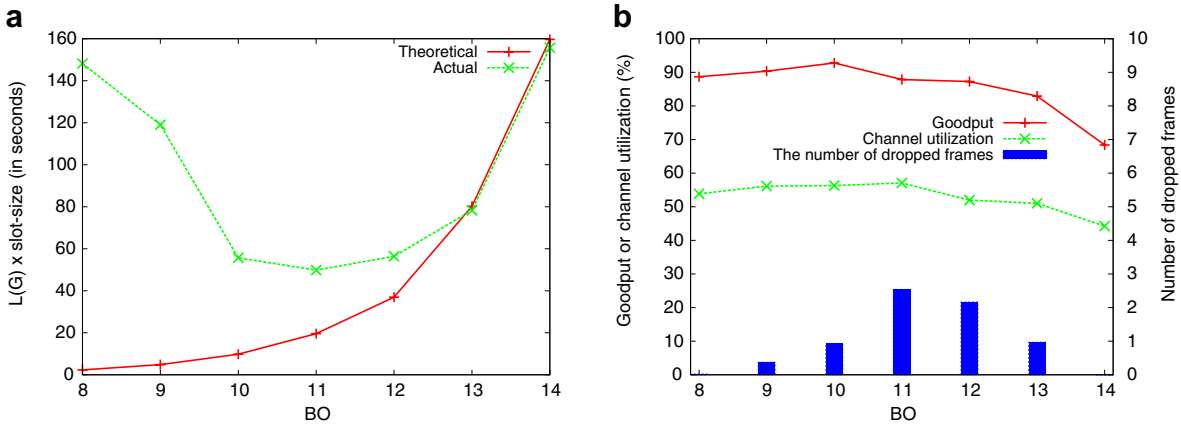

Fig. 9. Simulations considering buffer limitation and contention effects: (a) Theoretical vs. actual report latencies and (b) goodput, channel utilization, and number of dropped frames.

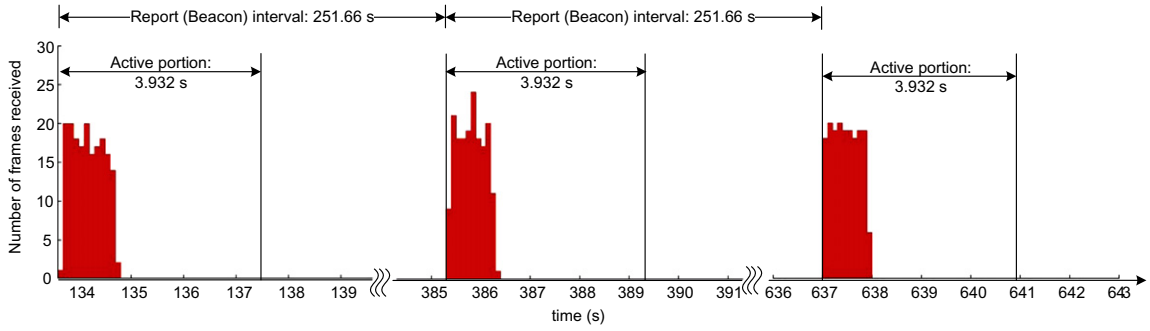

Fig. 10. A log of the number of frames received by a sink's child router when BO = 14.

BO is small, the report latency can not achieve to the theoretical value. This is because that an active portion is too small to accommodate all reports from sensors, thus lengthening the report latency. When BO becomes larger, the theoretical and actual curves would meet. However, the good put will degrade, as shown in Fig. 12(b). This is because reports are likely to be dropped due to buffer overflow. How to determine a proper BO, which can contain most of the reports and guarantee low latency, is an important design issue for such scenarios.

6. Conclusions

In this paper, we have defined a new minimum delay beacon scheduling (MDBS) problem for convergecast with the restrictions that the beacon scheduling must be compliant to the ZigBee standard. We prove the MDBS problem is NP-complete and propose optimal solutions for special cases and two heuristic algorithms for general cases. Simulation results indicate the performance of our heuristic algorithms decrease only when the number of interference

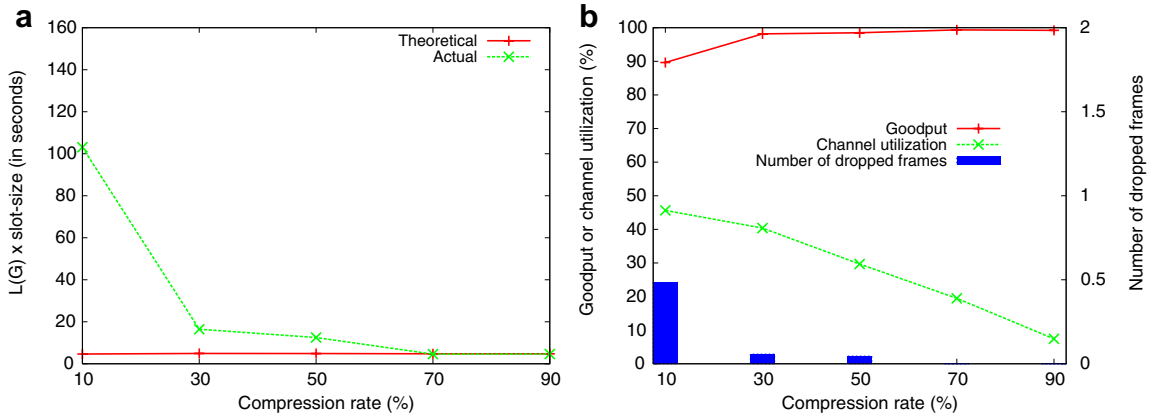

Fig. 11. Simulations considering data compression: (a) Theoretical vs. actual report latencies and (b) goodput, channel utilization, and number of dropped frames.

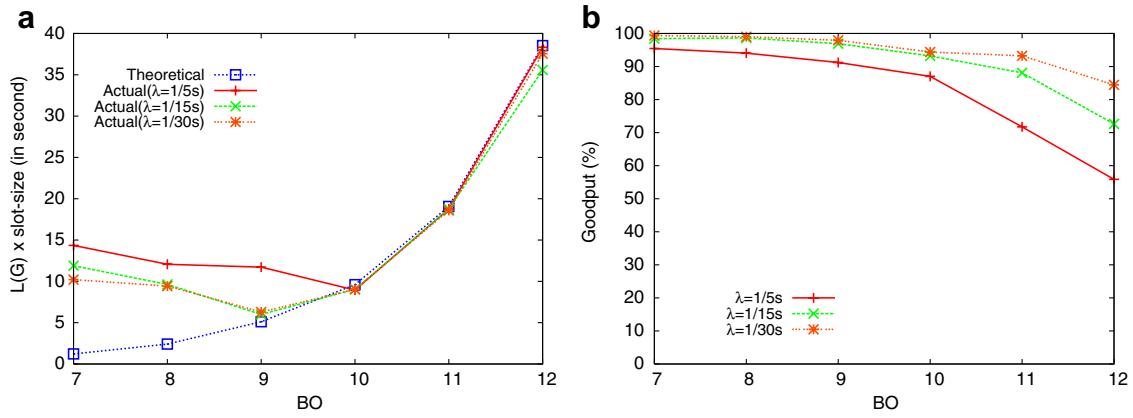

Fig. 12. Simulation results of event-driven scenarios: (a) theoretical vs. actual report latencies and (b) goodput.

neighbors is increased. Compared to the random slot assignment and greedy slot assignment scheme, our heuristic algorithms can effectively schedule the ZigBee routers' beacon times to achieve quick convergecast. In the future, it deserves to consider extending this work to an asynchronous sleep scheduling to support energy-efficient convergecast in ZigBee mesh networks.

Acknowledgements

Y.-C. Tseng's research is co-sponsored by Taiwan MoE ATU Program, by NSC Grants 93-2752-E-007-001-PAE, 96-2623-7-009-002-ET, 95-2221-E-009-058-MY3, 95-2221-E-009-060-MY3, 95-2219-E-009-007, 95-2218-E-009-209, and 94-2219-E-007-009, by Realtek Semiconductor Corp., by MOEA under Grant No. 94-EC-17-A-04-S1-044, by ITRI, Taiwan, by Microsoft Corp., and by Intel Corp.

References

[1] Chipcon CC2420DBK. Available from: <http://www.chipcon.com/>.
[2] Dust network Inc. Available from: <http://dust-inc.com/flash-index.shtml>.
[3] Design and construction of a wildfire instrumentation system using networked sensors. Available from: <http://firebug.sourceforge.net/>.
[4] Habitat monitoring on great duck island. Available from: <http://www.greatduckisland.net/technology.php>.
[5] Jennic JN5121. Available from: <http://www.jennic.com/>.
[6] Motes, smart dust sensors, wireless sensor networks. Available from: <http://www.xbow.com/>.
[7] S.-J. Baek, G. de Veciana, X. Su, Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation, IEEE Journal on Selected Areas in Communications 22 (6) (2004) 1130–1140.
[8] H. Choi, J. Wang, E.A. Hughes, Scheduling on sensor hybrid network, in: Proceedings of IEEE International Conference on Computer Communications and Networks (ICCCN), San Diego, USA, 2005.
[9] S. Gandham, Y. Zhang, Q. Huang, Distributed minimal time convergecast scheduling in wireless sensor networks, in: Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS), Lisboa, Portugal, 2006.
[10] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, J. Heidemann. An evaluation of multiresolution storage for sensor networks, in: Proceedings of ACM International Conference on Embedded Networked Sensor Systems (SenSys), Los Angeles, USA, 2003.
[11] B. Hohlt, L. Doherty, E. Brewer, Flexible power scheduling for sensor networks, in: Proceedings of ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Berkeley, USA, 2004.

[12] Y.-K. Huang, A.-C. Pang, T.-W. Kuo, AGA: adaptive GTS allocation with low latency and fairness considerations for IEEE 802.15.4, in: Proceedings of IEEE International Conference on Communications (ICC), Istanbul, Turkey, 2006.
[13] IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks specific requirements part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), 2003.
[14] IEEE standard for information technology – telecommunications and information exchange between systems – local and metropolitan area networks specific requirements part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs) (revision of IEEE Std 802.15.4-2003), 2006.
[15] Q. Li, M. DeRosa, D. Rus, Distributed algorithm for guiding navigation across a sensor network, in: Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Maryland, USA, 2003.
[16] C.-Y. Lin, W.-C. Peng, Y.-C. Tseng, Efficient in-network moving object tracking in wireless sensor networks, IEEE Transactions on Mobile Computing 5 (8) (2006) 1044–1056.
[17] G. Lu, B. Krishnamachari, C.S. Raghavendra, An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks, in: Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Mexico, USA, 2004.
[18] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, C.-F. Huang, Location tracking in a wireless sensor network by mobile agents and its data fusion strategies, The Computer Journal 47 (4) (2004) 448–460.
[19] Y.-C. Tseng, M.-S. Pan, Y.-Y. Tsai, Wireless sensor networks for emergency navigation, IEEE Computer 39 (7) (2006) 55–62.
[20] S. Upadhyayula, V. Annamalai, S.K.S. Gupta, A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks, in: Proceedings of IEEE Global Telecommunications Conference (Globecom), San Francisco, USA, 2003.
[21] D.B. West, Introduction to Graph Theory, Prentice Hall, 2001.
[22] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, S. Singh, Exploiting heterogeneity in sensor networks, in: Proceedings of IEEE INFOCOM, Miami, USA, 2005.
[23] Y. Yu, B. Krishnamachari, V.K. Prasanna, Energy-latency tradeoffs for data gathering in wireless sensor networks, in: Proceedings of IEEE INFOCOM, Hong Kong, 2004.
[24] ZigBee specification version 2006, ZigBee document 064112, 2006.