# Distributed Wake-up Scheduling for Data Collection in Tree-Based Wireless Sensor Networks

Fang-Jing Wu and Yu-Chee Tseng

***Abstract*—In a multi-hop wireless network, a conventional way of defining interference neighbors is to prohibit a node from using the same slot/code as those of its 1-hop and 2-hop neighbors. However, for data collection in a wireless sensor network, since the set of communication nodes is limited and the transmission directions are toward the sink, we show that a less strict set of interference neighbors can be defined. Based on this observation, we develop an efficient distributed wake-up scheduling scheme for data collection in a sensor network that achieves both energy conservation and low reporting latency.**

***Index Terms*—Communication protocol, power saving, sensor network, slot/code assignment, wireless network.**

## I. Introduction

COLLECTING sensing data is an important function of a wireless sensor network (WSN). It involves a subset of nodes, $\mathcal{R}$, each requested to report its sensory data via a *data collection tree* to the sink. Two main technical issues are *power saving* and *latency*. The former is to prolong network lifetime, while the latter concerns the freshness of data. To simultaneously address these two issues, we define a *wake-up scheduling problem*, where nodes can periodically switch between sleep and active modes. A node, if involved in the data collection tree, will receive an *active slot*. During its active slot, a node must wake up to collect data from its children. Then, it can go to sleep. Note that a node also needs to wake up to cooperate with its parent's active slot. On the other hand, for latency concern, data forwarding along the tree should be bounded. Besides, interference among these transmission activities should be avoided.

To avoid interference, a conventional way is to avoid a node from using the same slot as those used by its neighbors within two hops. However, in the data collection scenario in WSNs, since communication only involves partial nodes and the communication directions are always toward the sink along a data collection tree, the definition of interference can be relaxed. Motivated by this observation, this paper shows how to define the tightest set of interference neighbors when assigning active slots to nodes. Based on this definition, we then design an efficient distributed wake-up scheduling scheme for data collection in a WSN to meet the interference-free and low-latency requirements.

Manuscript received March 26, 2009. The associate editor coordinating the review of this letter and approving it for publication was X. Cao.

The authors are with the Department of Computer Science, National Chiao-Tung University, Taiwan (e-mail: fangjing@cs.nctu.edu.tw).

Y.-C. Tseng's research is co-sponsored by the MoE ATU Plan, by NSC grants 96-2218-E-009-004, 97-3114-E-009-001, 97-2221-E-009-142-MY3, and 98-2219-E-009-005, by MOEA 98-EC-17-A-02-S2-0048 and 98-EC-17-A-19-S2-0052, and by ITRI, Taiwan.

Digital Object Identifier 10.1109/LCOMM.2009.090712

Several efforts have focused on data collection in a WSN. In [1][2], nodes of the same depth in the data collection tree will have the same wake-up time. Work [1] proposes a *staggered wake-up scheme*, while [2] extends [1] to a *multi-parent scheme* such that a node can choose one parent with the earliest wake-up time to relay data. Unfortunately, [1][2] are not compatible with ZigBee and nodes of the same depth may suffer from interference. The work [3] proposes a ZigBee-compatible scheduling for convergecast, but it involves all nodes to report their data. It is a special case of our work and still adopts the conventional interference definition. Based on a TDMA model, [4] shows how to assign transmission slots to nodes to avoid interference. Reference [5] further improves [4] by reducing the latency when collecting data along the tree. Although a less strict definition of interference is used in [4] and [5], interference actually happens at the receivers' side. Our work does consider avoiding interference from this aspect and allows multiple transmitters to compete for a receiver at the latter's slot by following ZigBee's rules.

## II. Modeling Interference for Data Collection

A WSN is modeled as an undirected graph $G = (V, E)$, where $V$ contains all nodes and $E$ contains all communication links between nodes. One special node in $V$ is designated as the *sink*. A set of nodes $\mathcal{R} \subseteq V$ is requested to conduct *data collection* in the sense that each node needs to periodically send its sensing data to the sink, and these data may be aggregated on their way to the sink. Our goal is to construct a subtree $T$ from $G$ rooted at the sink connecting all nodes in $\mathcal{R}$ and schedule the wake-up time of nodes in $T$ for energy-saving and low-latency purposes. Note that $T$ is not necessarily a spanning tree of $G$.

We adopt a time-division model by dividing time into fixed-length slots. Each $k$ consecutive slots are grouped together and called a frame. In each frame, each node $v_i$ in $T$ will be assigned a wake-up slot $s_i \in \{0, 1, \ldots, k-1\}$. During slot $s_i$, $v_i$ must wake up to announce a beacon to synchronize with its children and then collect sensory data from them. Excluding $s_i$, $v_i$ may go to sleep. The value of $k$ should be large enough to ensure each node to find a slot.

The assignment of wake-up slots should meet two goals simultaneously: (i) the communication must be interference-free and (ii) the overall reporting latency from leaves of $T$ to the sink should be minimized. To address goal (i), one typical approach is to enforce a node not to use the same wake-up slot as any of its 1-hop and 2-hop neighbors. However, in our data collection scenario, since not all nodes are involved in the communication and communication directions are always toward the sink, a node only needs to consider a tighter set of interference neighbors, as defined below.

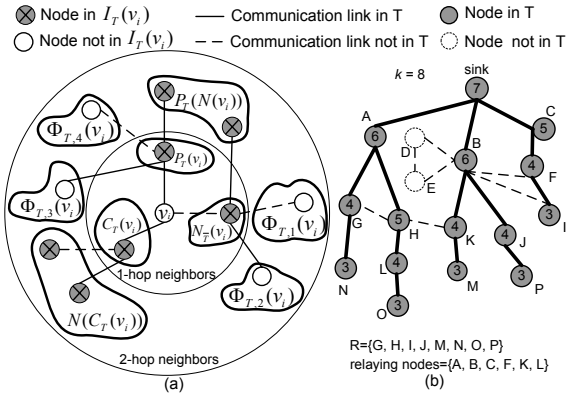


Fig. 1. (a) Classification of $v_i$'s 1-hop and 2-hop neighbors, and (b) an example of slot assignment, where the number in a circle is the node's slot.

**Definition 1.** *Given a node $v_i$ and a data collection tree $T$ in $G$, we define $P_T(v_i)$ as the set of $v_i$'s parent in $T$, $C_T(v_i)$ as the set of $v_i$'s children in $T$, $N(v_i)$ as the set of $v_i$'s neighbors in $G$, and $N_{\overline{T}}(v_i) = N(v_i) - P_T(v_i) - C_T(v_i)$ (i.e., $v_i$'s neighbors excluding $v_i$'s parent and children in $T$). We define the* interference set of $v_i$ with respect to $T$ *as:*

$$I_T(v_i) = N(v_i) \cup N(C_T(v_i)) \cup P_T(N(v_i)) - \{v_i\}. \quad (1)$$

Eq. (1) contains $v_i$'s direct interference set ($N(v_i)$) and indirect interference set ($N(C_T(v_i)) \cup P_T(N(v_i))$). The first set contains all $v_i$'s 1-hop neighbors. However, the second set may not contain all $v_i$'s 2-hop neighbors. Nodes in $v_i$'s 2-hop neighbors but not in $N(C_T(v_i)) \cup P_T(N(v_i))$ can be divided into four subsets:

$$\begin{aligned}
\Phi_{T,1}(v_i) &= N_{\overline{T}}(N_{\overline{T}}(v_i)) - I_T(v_i) \\
\Phi_{T,2}(v_i) &= C_T(N_{\overline{T}}(v_i)) - I_T(v_i) \\
\Phi_{T,3}(v_i) &= C_T(P_T(v_i)) - I_T(v_i) \\
\Phi_{T,4}(v_i) &= N_{\overline{T}}(P_T(v_i)) - I_T(v_i). \quad (2)
\end{aligned}$$

Fig. 1(a) shows how we divide $v_i$'s 1-hop and 2-hop neighbors in an abstract way. Considering node $K$, Fig. 1(b) shows an example, where $G \in \Phi_{T,1}(K)$, $L \in \Phi_{T,2}(K)$, $J \in \Phi_{T,3}(K)$, and $F \in \Phi_{T,4}(K)$. We see that data reporting from $N$ to $G$ and from $M$ to $K$ can coexist without interference. Similarly, reception at $L$, $J$, and $F$ is also interference-free. Note that interference should be decided at the receiver side, not the sender side. Therefore, all $G$, $L$, $J$, $F$, and $K$ can use the same slot (4). The following theorem proves that Eq. (1) gives the tightest interference set.

**Theorem 1.** *Given $\mathcal{R}$ and a tree $T$ in $G$, a slot assignment for data collection is interference-free iff for each pair of $v_i$ and $v_j$ such that $v_j \in I_T(v_i)$, we have $s_i \neq s_j$.*

*Proof:* To prove the *if* part, we will show that if an assignment achieves $s_i \neq s_j$ for each pair of $v_i$ and $v_j$ such that $v_j \in I_T(v_i)$ then the assignment is interference-free. Consider each $v_j$ in $v_i$'s 1-hop and 2-hop neighbors (refer to Fig. 1(a)). It is clear that no $v_j \in I_T(v_i)$ will cause interference with $v_i$. For each $v_j \in \Phi_{T,q}(v_i), q = 1 \ldots 4$, we will show that if $s_j = s_i$, the reception activities of $v_i$ and $v_j$ will not suffer from interference (there is no need to consider their transmission activities because this will be examined when considering their parents). Without loss of generality, we consider any child $v_c$ of $v_j$; there are two cases.

1. If $v_c$ is 1-hop away from $v_i$, then $v_j \in P_T(N(v_i)) \subseteq I_T(v_i)$, which implies $s_i \neq s_j$. So, $v_j$ will not choose the same slot as $v_i$.

2. If $v_c$ is 2-hop or above away from $v_i$, then $v_c$'s signal cannot be heard by $v_i$. So, $v_i$ will not be interfered by $v_c$'s transmission.

To prove the *only if* part, we show that if an assignment for $T$ is interference-free, for each pair $v_i$ and $v_j$ such that $v_j \in I_T(v_i)$, we have $s_i \neq s_j$. This part is proved by contradiction. Assume that there is a pair of $v_i$ and $v_j$ such that $v_j \in I_T(v_i)$ and $s_i = s_j$. By Eq. (1), $v_j$ may fall in three subsets.

1. If $v_j \in N(v_i)$, $s_i = s_j$ will lead to direct interference, which is a contradiction.

2. If $v_j \in N(C_T(v_i))$, $s_i = s_j$ will cause reception at the $v_j$ side being interfered by the transmission of $v_i$'s children, a contradiction.

3. If $v_j \in P_T(N(v_i))$, $s_i = s_j$ will cause reception at the $v_i$ side being interfered by the transmission of $v_j$'s children, a contradiction. ■

To address goal (ii), given a tree $T$, we then define the latency of data collection along $T$. The data collection latency $L_T(v_i, v_j)$ along a tree link $(v_i, v_j)$ is the number of slots from $v_i$ collecting a report from a child to $v_i$ forwarding them to $v_j$, i.e., $L_T(v_i, v_j) = (s_j - s_i) \mod k$. Similarly, the data collection latency along a tree path $P = v_{\alpha_1} \to v_{\alpha_2} \to \cdots \to v_{\alpha_m}$ is defined as $L_T(P) = L_T(v_{\alpha_1}, v_{\alpha_2}) + L_T(v_{\alpha_2}, v_{\alpha_3}) + \cdots + L_T(v_{\alpha_{m-1}}, v_{\alpha_m})$. Finally, the data collection latency of $T$ is defined as $L_T = \max\{L_T(P) |$ each path $P$ from a leaf of $T$ to sink$\}$. For example, in Fig. 1(b), $L_T(K, B) = 2$, $L_T(P = K \to B \to sink) = 3$, and $L_T = 4$.

**Definition 2.** *Given network $G = (V, E)$, data collection set $\mathcal{R}$, and $k$ available slots, the* wake-up scheduling problem *is to find a subtree $T$ in $G$ and a slot assignment for each node in $T$ which is interference-free and the overall reporting latency $L_T$ is as small as possible.*

## III. Distributed Wake-up Scheduling Algorithm

Based on the definition in Eq. (1), we propose a distributed algorithm. It contains two phases. The first *tree-formation phase* is to form a subtree $T$ to connect all nodes in $\mathcal{R}$. The second *slot-allocation phase* is to find an interference-free slot for each node in $T$ with low latency $L_T$.

**Tree-formation phase**: Each node attempts to join the network using its interference set as the metric. Note that first $T$ will include all nodes and then some nodes may truncate themselves from $T$ later on.

1. To initiate a new data collection task, the sink floods a *FORM_TREE($\mathcal{R}$)* packet to the whole network.

2. On receipt of the *FORM_TREE($\mathcal{R}$)* packet, each node $v_i$ will repeatedly broadcast a *HELLO* packet containing its parent (if it already has one) and its 1-hop neighbor set to its 2-hop neighbors for a period of time. It will also collect these information from its 2-hop neighbors.

3. Each node $v_i$ will try to find a node as it parent from those neighbors which have already joined $T$. If there are multiple

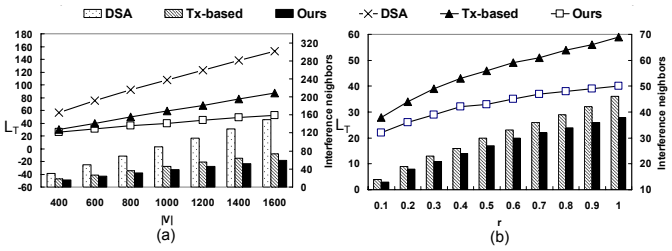


Fig. 2. (a) $L_T$ vs. $|V|$ ($k = 128$ slots). (b) $L_T$ vs. $r$ (n=1000).

candidates, $v_i$ will choose the one which causes the least value of $(depth(P_{T'}(v_i)), |I_{T'}(P_{T'}(v_i))|)$ as its parent. Note that $depth(P_{T'}(v_i))$ can help reduce the data collection latency and $|I_{T'}(P_{T'}(v_i))|$ can help improve slot reuse. Here we give priority to the former term. So, a pair $(a, b)$ is considered less than a pair $(c, d)$ if $a < c$ or $a = c$ but $b < d$. (Note that here we use $T'$ on purpose to reflect the fact that $T$ is now under construction. So, here $I_{T'}(v_i)$ means the set of interference nodes that $v_i$ knows so far, from overheard *HELLO*s.)

4. After $v_i$ chooses a parent node, there are two cases. (a) If $v_i \in \mathcal{R}$, it unicasts a *JOIN_TREE($v_i$)* packet to the sink. (b) If $v_i \notin \mathcal{R}$, it sets a timer $\Delta T_{join}$ and then waits for any *JOIN_TREE(·)* passing it. If it helps relay any *JOIN_TREE(·)*, it should join $T$. Otherwise, after $\Delta T_{join}$ expires, it truncates itself from $T$ by sending a *TRUNCATE* packet to its parent and flooding a *TRUNCATE* packet to its children. Note that the selection of $\Delta T_{join}$ depends on the expected depth of $T$. A too small $\Delta T_{join}$ will cause some nodes in $\mathcal{R}$ unable to join $T$. We suggest that $\Delta T_{join}$ should be proportional to the depth of $T$ plus some guard time.

5. After receiving *JOIN_TREE(·)* packets from all members of $\mathcal{R}$, the sink will flood an *ASSIGN_SLOT($\mathcal{R}$)* packet to the whole network to terminate this phase and start the next phase.

**Slot-allocation phase**: Nodes in $T$ will compete for slots in a top-down manner. A loop is used to select a node's slot, and priority is given to nodes with larger interference sets.

1. After sending out the above *ASSIGN_SLOT($\mathcal{R}$)* packet, the sink will assign slot $k-1$ to itself and repeatedly broadcast a *GET_SLOT*$(sink, k-1)$ packet to its 2-hop neighbors for a period of time.

2. When a node $v_i \in T$ without a wake-up slot receives a *GET_SLOT*$(v_j, s_j)$ from its parent node $P_T(v_i)$, it will set a tentative variable $t_i = (s_{P_T(v_i)} - 1)$.

3. Then, $v_i$ will try to find a candidate slot $s_i$ as follows. (a) Let $S \subseteq I_T(v_i)$ be the set of nodes in $v_i$'s interference set that have already decided their wake-up slots (this can be collected from *GET_SLOT(·)* packets). (b) If slot $(t_i \mod k)$ conflicts with any slot owned by nodes in $S$, decrement $t_i$ by 1 and repeat step (a); otherwise, go to step 4.

4. Node $v_i$ then sets $s_i = t_i \mod k$ as a candidate slot and repeatedly broadcasts a *COMPETE_SLOT*$(v_i, s_i)$ packet for a period of time $\Delta T_{compete}$. Here, $\Delta T_{compete}$ should be large enough for each node to disseminate/collect information to/from its 1-hop and 2-hop neighbors. We suggest that $\Delta T_{compete}$ should be proportional to the square of the maximum degree of the WSN.

5. During $\Delta T_{compete}$, if $v_i$ receives any *REJECT_SLOT($s_i$)* packet, $v_i$ must go back to step 3 by decrementing $t_i$ by 1 and find another candidate slot. After $\Delta T_{compete}$, if no *REJECT_SLOT(·)* packet is received, $v_i$ will confirm using $s_i$ by repeatedly broadcasting *GET_SLOT*$(v_i, s_i)$ packets to its 2-hop neighbors for a period of time.

6. When $v_i$ receives a *COMPETE_SLOT($v_j, s_j$)* packet from $v_j$ and $s_j = s_i$, $v_i$ has a higher priority over $v_j$ if (i) $v_i$ has already broadcasted *GET_SLOT($v_i, s_i$)* or (ii) $v_i$ has already broadcasted *COMPETE_SLOT($v_i, s_i$)* and $|I_T(v_i)| > |I_T(v_j)|$ or $(|I_T(v_i)| = |I_T(v_j)| \wedge (i > j))$. If so, $v_i$ unicasts a *REJECT_SLOT($s_j$)* to $v_j$; otherwise, $v_i$ will receive a *REJECT_SLOT(·)* packet in step 5 and then goes to step 3.

## IV. Simulation Results and Conclusions

We randomly deploy $n$ nodes in a $200 \times 200$ $m^2$ region. Each node has a transmission range of $20$ $m$, and there are $k = 128$ available slots. Let $r = \frac{|R|}{|V|}$ be the ratio of data collection nodes. Note that when $r = 1.0$, the problem is equivalent to the convergecast problem [3]. We compare our scheme against *DSA* [3] and *Tx-based scheme* [5], both of which use a BFS tree to connect all nodes. In DSA, each node has a reception slot by considering 2-hop neighbors as its interference set. In Tx-based scheme, each node has a transmission slot by considering its 1-hop neighbors, its siblings, and the children of its neighbors as its interference set. Fig. 2(a) compares $L_T$ (denoted by lines) under various numbers of $|V|$ when $r = 1.0$. Our scheme has around $62.60\%$ and $32.28\%$ less latency than DSA and Tx-based scheme, respectively. This is because the size of our interference set (i.e., $I_T(v_i)$) is only about $42.56\%$ and $83.12\%$ of that in DSA and in Tx-based scheme, respectively, as shown in Fig. 2(a) (denoted by bars). Fig. 2(b) investigates the impact of $r$ on $L_T$. It shows that $L_T$ will increase as $r$ increases, but our increasing rate, contributed by Eq. (1), is relatively slower.

To conclude, this paper considers a data collection scenario in WSNs which has less strict constraints on interference. We then propose an efficient scheduling and verify it via simulations. Our scheme can only handle one data collection task. When there are multiple tasks, one direct extension is to union all their sets $\mathcal{R}$s and regard them as one task. A future research direction is how to schedule multiple tasks at the same time in an efficient way.

## References

[1] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," in *Proc. IEEE Int'l Parallel and Distributed Processing Symp.*, 2004, pp. 224–231.

[2] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, 2006, pp. 322–333.

[3] M.-S. Pan and Y.-C. Tseng, "Quick convergecast in ZigBee beacon-enabled tree-based wireless sensor network," *Computer Comm.*, vol. 31, no. 5, pp. 999–1011, 2008.

[4] L. Paradis and Q. Han, "TIGRA: timely sensor data collection using distributed graph coloring," in *Proc. IEEE Int'l Conf. Pervasive Computing and Communications*, 2008, pp. 264–268.

[5] H. Wu, Q. Luo, and W. Xue, "Distributed cross-layer scheduling for in-network sensor query processing," in *Proc. IEEE Int'l Conf. Pervasive Computing and Communications*, 2006, pp. 180–189.