Two-Way Beacon Scheduling in ZigBee Tree-Based Wireless Sensor Networks

Lun-Wu Yeh[†], Meng-Shiuan Pan[†], and Yu-Chee Tseng^{†‡}

[†]Department of Computer Science

National Chiao Tung University, Taiwan

[‡]Department of Information and Computer Engineering

Chung-Yuan Christian University, Taiwan

Email: {lwye, mspan, yctseng}@cs.nctu.edu.tw

Abstract

Broadcast and convergecast are two fundamental operations in wireless sensor networks. Although previous works have addressed energy-efficient and low-latency scheduling, these works either consider only one-way (broadcast or convergecast) communication or are not compliant to ZigBee/IEEE 802.15.4 standards. Motivated by these observations, this work defines a two-way beacon scheduling problem for ZigBee tree-based networks. We propose a scheduling algorithm that can reduce indirect interference neighbors and achieve low-latency broadcast and convergecast. Simulation results are presented.

1. Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks* (WSNs) possible. A WSN normally consists of many inexpensive wireless nodes, each capable of collecting, processing, and storing environmental information, and communicating with neighboring nodes. Many WSN applications have been developed, such as emergency guiding [6][12], object tracking [10], and environmental monitoring [3][13].

Recently, several WSN platforms have been developed, such as MICAz [2], Tmote [4], and Dust Network [1]. To ensure interoperability of different platforms, the ZigBee/IEEE 802.15.4 [7][16] standards are proposed, which define physical, MAC, and network layers for low-rate, low-power wireless communications.

Sending commands to several nodes from the sink node (broadcast) or reporting sensory data to the sink node (convergecast) are two fundamental operations in WSNs. Broadcast and convergecast are inverse operation of each other. Some previous works [9][11][14] propose energy-

efficient and low-latency scheduling designs. Reference [11] aims to minimize the end-to-end communication delay while provides energy-efficient sleep mode for sensor nodes. Reference [9] analyzes different wake-up schemes on energy consumption and upstream/downstream delays. It is suggested that a node should wake up twice per cycle to support two-way transmission. The results in [9][11] are not compliant to the ZigBee/IEEE 802.15.4 standards. In [14], the authors propose convergecast algorithms which are compliant to ZigBee/IEEE 802.15.4. We propose an observation to reduce some indirect interferences in that network. It can decrease the transmission delay. Also, the broadcast issue is not addressed.

Our goal is to design an efficient beacon scheduling solution for ZigBee/IEEE 802.15.4 tree-based networks to support both broadcast and convergecast with low latencies in both directions. Fig. 1(a) shows the problem scenario. The network contains one sink (ZigBee coordinator), some *full-function* devices (ZigBee routers), and some *reduced-function* devices (ZigBee end devices). Each ZigBee router is responsible for collecting sensory data from end devices and relaying them to the sink. According to the IEEE 802.15.4 specification, a 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, a child device needs to wake up for an active portion and can communicate with its parent. However, to avoid collision with its neighbors, a router should shift its active portion by a certain amount. Fig. 1(b) shows a possible allocation of active portions for the sink c , routers R_A , R_B , and R_C . In this example, the sensory data reported from ED_1 and ED_2 can reach to the sink in one superframe. However, in the reverse direction, when the sink needs to send a packet to ED_1 , the latency will be up to 3 superframes. The transmission delay can not be negligible when the network is run under a low duty cycle. For example, in 2.4 GHz PHY, with 3.13% duty cycle, a superframe can be up to 251.658 seconds (with an active portion

Figure 1. (a) An example of ZigBee/IEEE 802.15.4 tree-based network. (b) An example of two-way transmission under the original ZigBee/IEEE 802.15.4 superframe structure. (c) An example of two-way transmission under the proposed modified ZigBee/IEEE 802.15.4 superframe structure.

of 7.88 seconds).

The above observation leads to the problem of designing two-way beacon scheduling for ZigBee tree-based networks such that the latencies of both broadcast and convergecast are as low as possible. We propose to modify the original superframe structure of IEEE 802.15.4 to allow each router to broadcast beacons twice in a superframe. This requires two active portions per superframe. One beacon is for the upstream (convergecast) direction and the other is for the downstream (broadcast) direction. Fig. 1(c) shows an allocation of active portions in the previous example. As can be seen, the transmission delays of downstream messages from the sink and the upstream reports from end devices ED_1 and ED_2 can be limited to one superframe.

Apparently, different assignment of active portions will incur different transmission delays. We propose an scheduling algorithm for low-delay, two-way (broadcast and convergecast) beacon scheduling. We show that assigning upstream and downstream active portions simultaneously can achieve lower delays than doing this separately. In addition, to further relieve the interference among neighboring routers, we propose a mechanism to reconnect end devices to different parent routers to reduce some indirect interferences. Simulation results show that the proposed algorithm can indeed achieve good performance.

The rest of this paper is organized as follows. Section 2

briefly introduces ZigBee/IEEE 802.15.4 standards and our proposed superframe structure. Section 3 formally defines the *two-way beacon scheduling (TBS)* problem. Section 4 presents our observations and algorithm for the TBS problem. Simulation results are given in Section 5. Finally, Section 6 concludes this paper.

2. Preliminaries

2.1. Overview of IEEE 802.15.4 and ZigBee

IEEE 802.15.4 [7] 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 MHz to 868.6 MHz, which provides a data rate of 20 kbps. Channels 1 to 10 work from 902.0 MHz to 928.0 MHz and each channel provides a data rate of 40 kbps. Channels 11 to 26 are located from 2.4 GHz 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 function devices (RFDs)*. IEEE 802.15.4 supports star and

Figure 2. The relationship between incoming and outgoing active portions.

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 becoming a coordinator or associating with an existing coordinator. A RFD can only associate with a coordinator.

The ZigBee alliance [5] defines the communication protocols above IEEE 802.15.4. In [16], star, tree, and mesh topologies are supported. A ZigBee coordinator is responsible for initializing, maintaining, and controlling the network. In a star network, devices must directly connect to the coordinator. For tree and mesh networks, devices can communicate with each other in a multihop fashion. The network backbone is formed by one ZigBee coordinator and multiple ZigBee routers (which must be 802.15.4 FFDs). RFDs can only join the network as end devices by associating with the ZigBee coordinator or ZigBee routers. 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.

The ZigBee coordinator defines the superframe structure of a network. As shown in Fig. 2, 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 portion, respectively. For a beacon-enabled network, the setting of BO and SO should satisfy the relationship $0 \leq SO \leq BO \leq 14$. (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 $1 \leq i \leq 16$. 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.

Figure 3. The proposed superframe structure for the TBS problem

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 [8], a router needs to select one active portion as its outgoing active portions, and based on the active portion selected by its parent, it also selects the same active portion as its incoming active portions (refer to Fig. 2). In an outgoing/incoming active portions, 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 active portion) should be shifted away from each other to avoid interference. However, the specification does not clearly define how to choose the locations of routers' active portions.

In this work, we consider two types 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 which has communication activities with one of these two routers. Both interferences should be avoided when choosing routers' active portions.

2.2. The Superframe Structure for Two-way Beacon Scheduling

We propose a new superframe structure to support quick broadcast and convergecast. In IEEE 802.15.4, each router should wake up in two slots (outgoing and incoming active portions). In this work, we propose that each router should wake up in four slots. These four slots are denoted as *TbUp*, *TbDn*, *RbUp*, and *RbDn*, as shown in Fig. 3. In *TbUp*/*TbDn* slots, a node will transmit beacons to its children for receiving upstream/transmitting downstream data from/to them. In *RbUp*/*RbDn* slots, a node will receive beacons from its parent for transmitting upstream/receiving downstream data

Figure 4. An example of two-way transmission delay.

to/from its parent. To support this modification, we can use the *reserved field* in the beacon frame to announce the positions of these extra slots. Except these changes, all other operations follow the original IEEE 802.15.4 specification. Hence, the new design is backward compatible with the original specification.

3. The Two-Way Beacon Scheduling Problem

We model the network as a graph $G = (V, E)$, where V contains all routers and the coordinator c and E contains all symmetric communication links between nodes in V . The coordinator and routers follow the superframe structure mentioned in Section 2.2. The coordinator serves as the sink. End devices are not included in G but will be associated to nodes in G . From G , we assume that a ZigBee tree $T = (V_T, E_T)$ has been constructed, where $V_T = V$ and $E_T \subseteq E$. Also, we can construct from G an *interference graph* $G_I = (V, E_I)$, where edge $(x, y) \in E_I$ if there are direct/indirect interferences between x and y . With parameters BO and SO , there are $k = 2^{BO-SO}$ active portions (slots) in a superframe. Motivated by Brook's theorem [15], which proves that n colors are sufficient to color any graph with a maximum degree of n , we assume that $k \geq 2 \times D_I$, where D_I is the maximum degree of G_I . The factor 2 is required here because each node will need up to 2 colors (slots).

To solve the broadcast and convergecast problems, each router $i \in V$ needs to decide two interference-free slots $s^u(i)$ and $s^d(i)$ among $[0, k-1]$ for its slots TbUp and TbDn, respectively. Its slots *RbUp* and *RbDn* will be the

slots $s^u(\text{par}(i))$ and $s^d(\text{par}(i))$, respectively, where $\text{par}(i)$ is the parent of i in tree T . By interference free, it implies that $s^u(i) \neq s^u(j)$, $s^u(i) \neq s^d(j)$, $s^d(i) \neq s^u(j)$, and $s^d(i) \neq s^d(j)$ if edge $(i, j) \in E_I$.

Our goal is to find a slot assignment such that the broadcast and convergecast latencies are as low as possible. Let (i, j) be a link in T such that i is the parent of j . The *upstream delay* $d^u(j, i)$ from j to i is the number of slots from j receiving a convergecast packet until it is forwarded to i , i.e.,

$$d^u(j, i) = (s^u(i) - s^u(j)) \bmod k.$$

Similarly, the *downstream delay* $d^d(i, j)$ from i to j is the number of slots from i receiving a broadcast packet until it is forwarded to j , i.e.,

$$d^d(i, j) = \begin{cases} 0, & \text{if } i \text{ is sink node} \\ (s^d(i) - s^d(\text{par}(i))) \bmod k, & \text{otherwise.} \end{cases}$$

Note that $d^d(i, j)$ is in fact independent on j 's selection because the transmission time of i is purely dependent on i 's selection. Also, the latency is 0 when i is the sink because the broadcast is initiated by i itself.

For any node i , the upstream delay from i to sink c , denoted by $D^u(i)$ is the sum of the per-hop upstream delays of the path from i to c on tree T . Similarly, for any node i , the downstream delay from sink c to i , denoted by $D^d(i)$ is the sum of per-hop downstream delays of the path from c to i . The overall delay incurred by T is defined as

$$L(T) = \max\{\max_{i \in V_T} D^u(i), \max_{j \in V_T} D^d(j)\}.$$

Figure 5. Simulation results on Observation 1.

Definition 1 Given a graph $G = (V, E)$, G 's interference graph $G_I = (V, E_I)$, a ZigBee tree $T = (V_T, E_T)$, and k available slots, two-way beacon scheduling (TBS) problem is to find an interference-free slot assignment $s^u(i)$ and $s^d(i)$ for each $i \in V$ such that network latency $L(T)$ is minimized.

Fig. 4 shows an example, the per-hop upstream delay $d^u(R_B, R_A)$ from router R_B to router R_A is 7 and the per-hop downstream delay $d^d(R_A, R_B)$ from router R_A to router R_B is 3. The upstream delay from R_D to c is 17 and the downstream delay from c to R_D is 11. The overall latency $L(T)$ of this example is 11.

4. The Proposed Scheme

In this section, we propose a centralized slot assignment algorithm for the TBS problem. We first present some observations.

Observation 1: Assigning upstream and downstream slots simultaneously can achieve lower delay than assigning upstream and downstream slots separately.

This observation is supported by simulations. The first slot assignment scheme (SA1) examines nodes of T in a bottom-up manner. For each node being visited, it will greedily pick a slot for its TbUp and a slot for its TbDn such that the per-hop latencies are smallest. This is repeated until the sink node is reached. The second slot assignment scheme (SA2) also visits nodes of T to the sink node in a bottom-up manner, but only assigns slots for TbUp. Then it visits T in a top-down manner and assigns slots for TbDn. Our simulation result is in Fig. 5, which shows that SA1 can achieve lower latency than SA2 in most cases.

Observation 2: Suppose that routers R_i and R_j have indirect interference because they have some common end devices. If some end devices can be reconnected, then the

Figure 6. (a) Original slot assignment. (b) The slot assignment after reconnecting ED_1 and ED_2 to R_3 .

indirect interference relation between R_i and R_j may be removed.

The common end devices of routers R_i and R_j must be located in the overlapping area of the transmission ranges of R_i and R_j . Fig. 6(a) shows an example where R_1 and R_2 have indirect interference because of the existence of end devices ED_1 and ED_2 . Given the slot assignment in the parentheses, the latency $L(T)$ is 3. If ED_1 and ED_2 are associated to c or R_3 , R_1 and R_2 will have no indirect interference. Then, as shown in Fig. 6(b), R_1 and R_3 can use the same slot pair (1, 6) to achieve a lower $L(T)$ of 2.

Observation 3: When selecting slots, a router with more interference neighbors should select its slots earlier.

A node with higher interference relation has less choices, so it should pick its slots earlier. This leads to Observation 3.

Below, we propose a centralized slot assignment algorithm for the TBS problem. We traverse routers in a bottom-up fashion according to their depths in T . For those 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 being visited, run the following two procedures: FindUpSlot(v) and FindDnSlot(v).

FindUpSlot(v)

1. For each v , a tentative variable $t^u(v)$ will be computed, from which the final slot $s^u(v)$ will be determined.
 - (a) If v is a leaf node, we set $t^u(v) = 0$.
 - (b) If v is a non-leaf node, we set $t^u(v) = \max\{s^u(v') | v' \text{ is a child of } v\} + 1$.
2. Let $N(v)$ be the set of nodes that have direct or indirect interference with v and have received slots, i.e., $N(v) = \{v' | (v', v) \in E_I \text{ and } s^u(v') \neq NULL\}$.

3. We will check if $temp = t^u(v) \bmod k$ is a feasible slot for v by examining the slots used by nodes in $N(v)$. There are two cases:

- (a) If there exists a $v' \in N(v)$ such that v' and v have direct interference and $s^u(v') = temp$ or $s^d(v') = temp$, then the slot $temp$ is not feasible for v .
- (b) If there exists a $v' \in N(v)$ such that v' and v have indirect interference and $s^u(v') = temp$ or $s^d(v') = temp$, then we will call procedure $RemoveIndInt(v', v)$ to see if such interference can be removed. If so, $temp$ is a feasible slot for v . Otherwise, $temp$ is not feasible for v .

If the above examination determines that $temp$ is a feasible slot for v , then we assign $s^u(v) = temp$. otherwise, we increase $t^u(v)$ by one and repeat step 3 again.

FindDnSlot(v)

1. For each v , a tentative variable $t^d(v)$ will be computed, from which the final slot $s^d(v)$ will be determined.
 - (a) If v is a leaf node, we set $t^d(v) = k - 1$.
 - (b) If v is a non-leaf node, we set $t^d(v) = \min\{s^d(v') | v' \text{ is a child of } v\} - 1$.
2. Let $N(v)$ be the set of nodes that have direct or indirect interference with v and have received slots, i.e., $N(v) = \{v' | (v', v) \in E_I \text{ and } s^d(v') \neq NULL\}$.
3. We will check if $temp = t^d(v) \bmod k$ is a feasible slot for v by examining the slots used by nodes in $N(v)$. There are two cases:
 - (a) If there exists a $v' \in N(v)$ such that v' and v have direct interference and $s^u(v') = temp$ or $s^d(v') = temp$, then the slot $temp$ is not feasible for v .
 - (b) If there exists a $v' \in N(v)$ such that v' and v have indirect interference and $s^u(v') = temp$ or $s^d(v') = temp$, then we will call procedure $RemoveIndInt(v', v)$ to see if such interference can be removed. If so, $temp$ is a feasible slot for v . Otherwise, $temp$ is not feasible for v .

If the above examination determines that $temp$ is a feasible slot for v , then we assign $s^d(v) = temp$. Otherwise, we decrease $t^d(v)$ by one and repeat step 3 again.

RemoveIndInt(v', v)

1. Indirect interference appears when there is a common neighbor x of v' and v . If x is a router, it can be ignored. If x is an end device, then we may try to associate x with another router, say w , other than v' and

v . If w already has its slots $s^u(w)$ and $s^d(w)$, then we have to make sure this does not cause new interference. Otherwise, x can be associated w since the interference, if any, can be resolved later on.

2. If for each common neighbor x of v' and v , the above step 1 allows us to reassociate x to another router to remove the interference, then a positive response will be replied by this procedure; otherwise, a negative response will be replied.

Note that each end device maintains a *potential parent routers* which contains routers that can connect the end device. Fig. 7(a) shows an example of this algorithm. We choose tentative slots (1, 15) for router R_6 . Routers R_6 and R_8 have common end devices ED_1 and ED_2 . After performing procedure $RemoveIndInt(v', v)$, ED_1 reconnects to other router. R_6 and R_8 are removed from potential parent routers of ED_1 and ED_2 . R_6 can use slots (1, 15), as in Fig. 7(b). Fig. 7(c) shows the result of the slot assignment algorithm, where ED_2 and ED_3 also change their parent routers. The result $L(T)$ is 5.

The computational complexity of this algorithm is analyzed below. Before three procedures, the complexity of sorting is $O(D_T \log D_T)$, where D_T is the maximum degree of T . In $RemoveIndInt(v', v)$, the time complexity is $O(D_I + D_E)$, where D_E is the maximum number of common end devices. In $FindUpSlot(v)$, the computational cost of computing tentative slot in step 1 is $O(D_T)$. In step 2 and 3 of $FindUpSlot(v)$, the time complexity for checking interference and call procedure $RemoveIndInt(v', v)$ is $O(D_I(D_I + D_E))$. The time complexity of $FindDnSlot(v)$ is the same as $FindUpSlot(v)$. And $FindUpSlot(v)$ and $FindDnSlot(v)$ will be executed $|V|$ times. Hence, the overall time complexity is $O(|V|D_T \log D_T + |V|D_I^2 + |V|D_I D_E)$.

5. Simulation Results

In this section, we use simulation programs to evaluate the proposed algorithm. In order to observe the effects of procedure $RemoveIndInt(v', v)$, we compare our scheme, denoted as SA, against a reduced version of SA, which does not reconnect end devices, denoted as SA-NR. Besides, we compare the SA against a greedy slot assignment algorithm, denoted as GSA, greedily chooses TbUp slots in a bottom-up manner to minimize per hop latency of upstream, and then chooses TbDn slots in a top-down manner to minimize per hop latency of downstream. In our simulations, routers and end devices are randomly distributed in a $N \times N$ region and a sink node is placed in the center of the network.

Fig. 8(a) shows the effects on the different network size. The number of routers and end devices are set to $(N/10)^2$

Figure 7. (a) In the process of the slot assignment algorithm. (b) After procedure RemoveIndInt(v', v), router R_6 and R_8 can use the same slots. (c) The result of the slot assignment algorithm.

and $(N/10)^2 \times 3$, respectively, and set $k = 128$. SA outperforms others. From the result, we can see that the procedure RemoveIndInt(v', v) can effectively reduce $L(T)$. GSA has the worst performance. This result corresponds to the Observation 1 on Section 4.

Next, we simulate a $300m \times 300m$ network, place 900 routers and 2700 end devices, and set $k = 128$. Fig. 8(b) shows that the result when we vary the transmission range. Because a larger transmission range implies higher interference neighbors. The latency $L(T)$ of three algorithms are increasing with transmission range. Due to the procedure RemoveIndInt(v', v) can effectively reduce indirect interferences, the latency $L(T)$ of SA slightly increases.

With a network size of $200m \times 200m$ and a router transmission range of $20m$, and $k = 128$, we vary the number of routers in the network. As Fig. 8(c) shows, when there are more and more routers, the number of interferences will increase. The latency $L(T)$ of GSA and SA-NR markedly increase, but SA does not. This is also because the procedure RemoveIndInt(v', v) can effectively reduce indirect interferences. In Fig. 8(d), we fix the number of routers and end devices to 900 and 2700, respectively, and vary routers' duty cycle. Note that a lower duty cycle means a larger number of available slots. In GSA and SA-NR, when available slots are $k = 2^6$, routers use up a round of slots quickly. So, these $L(T)$ is larger than others. Interestingly, when available slots are enough, the latency $L(T)$ of these algorithms are independent of the number of slots.

6. Conclusions

In this paper, we modify the original superframe structure and define a two-way beacon scheduling (TBS) problem for supporting two-way transmission in ZigBee tree-based networks. We propose a centralized algorithm to solve the TBS problem. Simulation results indicate that the proposed scheme can effectively reduce the number of interference neighbors and thus decrease the network latency. In the future, we would design a distributed version of slot assignment algorithm to support two-way transmissions.

7. Acknowledgements

Y.-C. Tseng's research is co-sponsored by Taiwan MoE ATU Program, by NSC grants 93-2752-E-007-001-PAE, 95-2221-E-009-058-MY3, 95-2221-E-009-060-MY3, 96-2219-E-009-007, 96-2218-E-009-004, 96-2622-E-009-004-CC3, and 96-2219-E-007-008, by Realtek Semiconductor Corp., by MOEA under grant number 94-EC-17-A-04-S1-044, by ITRI, Taiwan, by Microsoft Corp., and by Intel Corp.

References

- [1] Dust network inc. <http://www.dustnetworks.com/flash-index.shtml>.
- [2] Micaz mote. <http://www.xbow.com/Products/productdetails.aspx?sid=164>.

Figure 8. Simulation results on the network latencies under different configurations.

- [3] Terrestrial ecology observing systems. <http://research.cens.ucla.edu/areas/2007/Terrestrial/>.
- [4] Tmote sky. <http://www.moteiv.com/products/tmotesky.php>.
- [5] ZigBee alliance. <http://www.zigbee.org/>.
- [6] M. A. Batalin, G. S. Sukhatme, and M. Hattig. Mobile robot navigation using a sensor network. In *IEEE International Conference on Robotics and Automation*, 2004.
- [7] 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.
- [8] 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.
- [9] A. Keshavarzian, L. V. H. Lee, K. C. D. Lal, and B. Srinivasan. Wakeup scheduling in wireless sensor networks. In *Proc. of ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2006.
- [10] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. on Mobile Computing*, 5(8):1044–56, 2006.
- [11] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel. Delay efficient sleep scheduling in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2005.
- [12] M.-S. Pan, C.-H. Tsai, and Y.-C. Tseng. Emergency guiding and monitoring applications in indoor 3d environments by wireless sensor networks. *International Journal of Sensor Networks (IJSNet)*, 1(1/2):2–10, 2006.
- [13] R. Szweczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.
- [14] Y.-C. Tseng and M.-S. Pan. Quick convergecast in zigbee beacon-enabled tree-based wireless sensor networks. In *Computer Communications*, to appear.
- [15] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.
- [16] ZigBee specification version 2006, ZigBee document 064112, 2006.