WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
*Wirel. Commun. Mob. Comput.* 2013; **13**:47–62
Published online 18 January 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.1094

RESEARCH ARTICLE

# A lightweight, self-adaptive lock gate designation scheme for data collection in long-thin wireless sensor networks

You-Chiun Wang[1]*, Che-Hsi Chuang[1], Yu-Chee Tseng[1] and Chien-Chung Shen[2]

[1] Department of Computer Science, National Chiao-Tung University, Hsinchu 30010, Taiwan
[2] Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, U.S.A.

## ABSTRACT

Constrained by the physical environments, the *long-thin* topology has recently been promoted for many practical deployments of wireless sensor networks (WSNs). In general, a long-thin topology is composed of a number of long branches of sensor nodes, where along a branch each sensor node has only one potential parent node toward the sink node. Although data aggregation may alleviate excessive packet contention, the maximum payload size of a packet and the dynamically changing traffic loads may severely affect the amount of sensor readings that may be collected along a long branch of sensor nodes. In addition, many practical applications of long-thin WSNs demand the exact sensor readings at each location along the deployment areas for monitoring and analysis purposes, so sensor readings may not be aggregated when they are collected. This paper proposes a lightweight, self-adaptive scheme that designates multiple collection nodes, termed *lock gates*, along a long-thin network to collect sensor readings sent from their respective upstream sensor nodes. The self-adaptive lock gate designation scheme balances between the responsiveness and the congestion of data collection while mitigating the funneling effect. The scheme also dynamically adapts the designation of lock gates to accommodate the time-varying sensor reading generation rates of different sensor nodes. A testbed of 100 Jennic sensor nodes is developed to demonstrate the effectiveness of the proposed lock gate designation scheme. 

**KEYWORDS**

data collection; lock gates; long-thin networks; wireless sensor networks

***Correspondence**

You-Chiun Wang, Department of Computer Science, National Chiao-Tung University, Hsinchu 30010, Taiwan
E-mail: wangyc@cs.nctu.edu.tw

## 1. INTRODUCTION

A *wireless sensor network* (WSN) consists of a sheer number of sensor nodes, where each sensor node is a wireless device that reports sensor readings of its surroundings to a sink node *via* multi-hop *ad hoc* communications. Such networks facilitate pervasive monitoring of the physical environments to enable applications such as habitat monitoring, smart home, and surveillance [1–3].

In recent research, the *long-thin* topology has been promoted for many practical applications of WSNs where the sensor deployment is subject to environmental constraints [4]. For instance, a surveillance system of moving cars along streets, a monitoring system of carbon dioxide inside tunnels, and a monitoring system of water quality within underground sewer lines are typical applications. Figure 1(a) depicts a physical deployment of sensor nodes along streets, and Figure 1(b) depicts its corresponding long-thin network topology. In general, a long-thin topology is formed by a bunch of long *branches*, and each branch may be composed of tens or even hundreds of nodes. The structure of a branch is recursively defined, where a branch may contain other (sub)branches. For each sensor node along a branch, there exists only one potential parent node toward the sink node. Branches are grafted at *branch nodes*, and Figure 1(b) shows an example where a branch node is denoted with double circles.

Let $\{s_1, s_2, \cdots, s_N\}$ denote the node IDs of a long-thin WSN, where $N$ is the number of sensor nodes. By viewing a long-thin topology as a shortest-path tree rooted at the sink node, let $d_i$ be the depth of sensor node $s_i$ from the sink node. Assuming that each sensor node in the long-thin WSN delivers one sensor reading to the sink node without aggregating data, the network will incur $\sum_{i=1}^{N} d_i$ transmissions to collect all of the sensor readings. For instance, the long-thin WSN of Figure 1 incurs 62

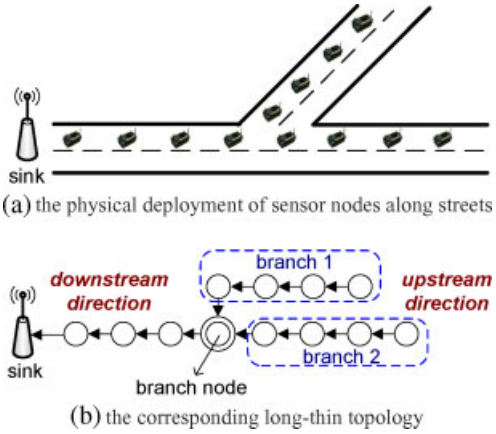



**Fig. 1.** An example of long-thin WSNs.

transmissions without data aggregation. Even worse, a long-thin WSN may suffer from the funneling effect where the hop-by-hop traffic over a long branch results in an increase in transit traffic intensity, collision, congestion, packet loss, and energy drain as packets move closer toward the sink node [5].

Now consider the following collection scheme. The leaf nodes start transmitting their sensor readings first. Each intermediate node *waits* to collect both its own sensor reading and the collected sensor readings sent from its children (or upstream nodes), and then forwards the collected packet toward the sink node. Such a scheme may result in only $N$ transmissions in the network, assuming that successively collected sensor readings could be loaded into one huge packet. For instance, the long-thin WSN of Figure 1 may incur only 12 transmissions using such a collection scheme. Fewer transmissions benefit WSNs by reducing contention and conserving energy.

Although the above collection scheme maximally reduces the number of transmissions in a long-thin WSN, constraints imposed by the maximum payload size $L_{\max}$ of a packet and the compression ratio $\delta$ ($0 < \delta \leq 1$) make such a scheme impractical for a long-thin topology where a node can only collect up to $\lfloor \frac{L_{\max}}{\delta} \rfloor$ bytes of sensor readings, while the total data size of sensor readings generated by sensor nodes along a long branch can easily exceed this bound. Besides, many practical applications of long-thin WSNs demand the exact sensor readings of each location along the deployment areas for data monitoring and analysis purposes, so that sensor readings are not aggregated (such as computing the average, the minimum, or the maximum value) while they are being collected.

To mitigate the funneling effect and to comply with the maximum payload size, this paper suggests that in a long-thin network, multiple collection nodes, termed *lock gates*,[†]

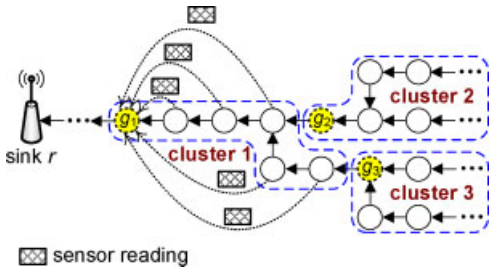



**Fig. 2.** A long-thin WSN and its lock gates.

should be designated, where each lock gate collects both its own sensor readings and all of the sensor readings sent from its upstream sensor nodes (up to immediate, upstream lock gates or the end of the branches) subject to the $\lfloor \frac{L_{\max}}{\delta} \rfloor$ bound. For instance, as shown in Figure 2, assuming $L_{\max}$ is 3 bytes, $\delta$ is 0.5, and the size of sensor readings is 1 byte, we designate one lock gate every six sensor nodes so that lock gate $g_1$ collects the sensor readings of itself and its five upstream nodes into one collected packet.[‡] Similar collections are done by lock gates $g_2$ and $g_3$. Such a collected packet (of maximum payload size $L_{\max}$) is said to be completely *filled up* with sensor readings and thus can be forwarded to the sink node without being padded with any other sensor readings along the way.

Another benefit of the above lock gate designation scheme is that it actually reduces contentions by spatially separating areas where packets are transmitted. In the above example, since lock gates $g_2$ and $g_3$ may *hold* their respective transmissions until enough sensor readings from the corresponding upstream nodes have been collected, the sensor nodes headed by lock gate $g_1$ can transmit their sensor readings to $g_1$ with less or even no interference coming from sensor nodes headed by lock gates $g_2$ and $g_3$.

Since sensor nodes may generate sensor readings at different *rates*, each lock gate may wait for a different amount of time to completely fill up one collected packet of payload size $L_{\max}$. Let $\lambda_j$ denote the sensor reading generation rate (in bytes/second) of sensor node $s_j$. The rate $\lambda_j$ may vary over time, but is assumed to change slowly so that a steady value of $\lambda_j$ could be observed over a short period of time. Let $C(g_i)$ denote the *cluster* of nodes containing lock gate $g_i$ and its upstream sensor nodes, up to the immediate, upstream lock gates or the end of the branches. Given $\lambda_j$ for each sensor node $s_j$ in $C(g_i)$, the expected time $T_i$ that lock gate $g_i$ takes to completely fill up one packet of maximum payload size $L_{\max}$ should satisfy:

$$\left( T_i \times \sum_{s_j \in C(g_i)} \lambda_j \right) \times \delta = L_{\max} \tag{1}$$

[†] We are inspired by the lock gates used in canals to withstand the water pressure arising from the level difference between adjacent pounds. In the context of long-thin WSNs, (irregular) network traffic from sensor readings corresponds to water pressure, which should be regulated to mitigate funnelling effect, for instance.

[‡] For the ease of explanation, in this example we simply assume that there is no 'protocol overhead' (or packet header/trailer) so that the size of a packet is exactly the same as the size of its payload.

When $T_i$ is large, the sink node is expected to wait for a longer time to receive a collected packet from lock gate $g_i$, which increases the application's response time. A large $T_i$ may imply either the size of $C(g_i)$ is small or the total sensor reading generation rate of the sensor nodes in $C(g_i)$ is low, or both. On the other hand, a small $T_i$ may imply either the size of $C(g_i)$ is large or the total sensor reading generation rate of the sensor nodes in $C(g_i)$ is high, or both. This may result in too many packet transmissions and thus congest the network. Therefore, lock gate designation should be made 'self-adaptive,' where $T_i$ is bounded by dynamically designating the positions of lock gates to balance between the response time and the congestion of data collection. Furthermore, since sensor nodes are usually with limited computation power and small memory size [6], the self-adaptive lock gate designation scheme should be lightweight with simple operations.

This paper proposes a *lightweight, self-adaptive lock gate designation scheme*, termed *ALT*, to facilitate effective data collection in long-thin WSNs. Given a pair of time thresholds ($T_{\min}$, $T_{\max}$), ALT adaptively designates lock gates in a long-thin WSN such that for each lock gate $g_i$, the condition $T_{\min} \leq T_i \leq T_{\max}$ holds. The thresholds $T_{\min}$ and $T_{\max}$ are specified by the application of a long-thin WSN to avoid congestion from transmitting excessive packets and to impose an upper bound on response time, respectively. The ALT scheme possesses three characteristics. First, sensor nodes do not need to report their sensor reading generation rates $\lambda_j$ to their corresponding lock gates. Instead, lock gates only need to *locally* observe their $T_i$ values for the execution of ALT, so that ALT has low message overhead and good response time. The experimental results in subsection 5.2 indeed show that ALT performs well even when $\lambda_j$ changes. Second, ALT adopts a simple scheme to move lock gates in a 'hop-by-hop' manner so that the operations are light-weight and can be easily implemented on practical sensor platforms. Third, in contrast to conventional clustering protocols designed for WSNs with random topology, ALT incurs much less control overhead in long-thin WSNs because clusters are formed automatically whenever a lock gate and its upstream lock gates are designated. In addition, lock gates do not need to know the identities of their respective cluster members, and sensor nodes do not need to send their sensor reading generating rates to their corresponding lock gates.

The contributions of this paper are three-fold. First, we point out the necessity of lock gates to balance between the response time and the congestion of data collection in a long-thin WSN, and propose a lightweight, self-adaptive scheme to designate the lock gates. To the best of our knowledge, ALT is the first effort addressing efficient data collection in long-thin WSNs. Second, to evaluate its performance, we implement ALT on sensor nodes equipped with the Jennic wireless micro-controller supporting the IEEE 802.15.4 protocol [7], and develop a testbed containing 100 sensor nodes. Experimental results demonstrate that ALT adapts well to varying sensor reading generation rates and incurs fewer message transmissions than other schemes. In addition, we show the benefits of lock gates which significantly reduce both the number of data retransmissions and the amount of packet losses at the MAC layer. Third, since placing lock gates may increase the packet latency of sensor nodes, we derive the average extra packet latency caused by a lock gate *via* mathematical analysis. We also validate the correctness of our analysis with measurements from real experiments.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives our problem statement. ALT and the analysis of its expected extra packet latency are described in Section 4. Section 5 presents our prototyping efforts and discusses experimental results. Conclusions are drawn in Section 6.

## 2. RELATED WORK

Long-thin WSNs are widely used in many monitoring applications such as leakage detection within fuel pipes, stage measurements inside sewer, traffic adjustment along tunnels or highways, vibration detection of bridges, and flood protection of rivers. The long-thin topology is first studied in Reference [4], which considers how to assign network addresses to sensor nodes in ZigBee-based long-thin WSNs to facilitate routing. In addition, a long-thin WSN is deployed along a river to monitor its water level. The work in Reference [8] discusses how to detect and correct localization errors in a long-thin WSN. In addition, it adopts a weighted voting scheme to detect possible faulty sensor readings. However, none of the existing work on long-thin WSNs addresses the issue of data aggregation/compression.

The subject of data aggregation/compression in WSNs with random topologies has been extensively studied in the literature. Below, we categorize and review existing solutions.

### 2.1. Tree-based aggregation

The objective of tree-based aggregation schemes is to maximize a WSN's lifetime by jointly optimizing data aggregation and routing tree formation [9]. For instance, the work of Kalpakis et al. [10] discusses how to find a set of data aggregation schedules to maximize the system's lifetime, where a *schedule* is defined as a collection of spanning trees rooted at the sink node. The work of Krishnamachari et al. [11] Intanagonwiwat et al. [12] proposes a data-centric approach to select an appropriate routing path to reduce energy consumption. TAG [13] organizes a WSN into a tree and proposes SQL-like semantics to aggregate streaming data into histograms. The work of Harris et al. [14] builds an aggregation tree according to the energy consumption of sensor nodes. Each node predicts the energy consumption of its potential parents and selects the one that can be left with the most energy as its parent. In the work of von Rickenbach and Wattenhofer [15], one coding tree for raw

data aggregation and one shortest-path tree for transmitting compressed data are built to deliver data to the sink node. In the work of Pattem et al. [16], the effect of data aggregation on different routing schemes is studied. The work also proposes a static clustering scheme to achieve a near-optimal performance for various spatial correlations. The above research efforts focus on how to choose a good routing metric based on data attributes to facilitate aggregation. However, in long-thin WSNs, there usually exists at most one route from a sensor node to the sink node (i.e., each sensor node has at most one potential parent node toward the sink node), so these existing tree-based solutions may not be directly applied.

### 2.2. Clustering-based aggregation

This category of schemes first group sensor nodes into clusters and then perform data aggregation within each cluster. The critical issue is how to select the *cluster head* in each cluster to aggregate data for its cluster members [17]. For instance, the scheme in Reference [18] assigns weights to each node and the nodes with larger weights may become cluster heads, while the work of Kuhn et al. [19] favors nodes with more neighbors (i.e., higher degrees) to become cluster heads. LEACH [20] assumes that each sensor node can be reachable in one hop and then assigns a fixed probability for each node to elect itself as a cluster head. In HEED [21], each sensor node uses its residual energy as the parameter to probabilistically select itself as a cluster head. The above research efforts discuss how to organize clusters such that nodes within a cluster are one-hop or $k$-hop away from the cluster head. In addition, SCT [22] proposes a ring-sector division clustering scheme, where sensor nodes in the same section are assembled into one cluster. Clearly, this ring-based approach cannot be used for long-thin WSNs. In contrast, ALT adaptively adjusts the size of clusters according to the amount of traffics generated from sensor nodes.

### 2.3. Chain-based aggregation

In these schemes, sensor nodes are organized into a linear chain for data aggregation. For instance, PEGASIS [23] organizes such a chain by adopting a greedy algorithm, where each sensor node selects its nearest neighbor (closer to the sink) as its *successor* along the chain and then sends its sensing readings to the successor. However, PEGASIS may not guarantee to minimize the total energy consumption of sensor nodes. Therefore, the work of Du et al. [24] proposes a chain-construction scheme that minimizes the total energy consumption of sensor nodes by reducing the value of $\sum D^2$, where $D$ is the distance between any two adjacent sensor nodes along the chain. The chain-based topology can be viewed as one special instance of the long-thin topology. Nevertheless, in the chain-based aggregation schemes, except for the node(s) at the end of the chain, all the sensor nodes along the chain act as aggregators. In contrast, ALT dynamically selects a subset of sensor nodes to act as lock gates according to the sensor reading load.

### 2.4. Hierarchical aggregation

Several studies adopt a hierarchical architecture to aggregate or compress data in a WSN. The work of Chen et al. [25] first selects a subset of sensor nodes as level-1 aggregators. Then, among these level-1 aggregators, the scheme selects a subset of level-1 aggregators to act as level-2 aggregators. This procedure is repeated until level-$h$ aggregators are selected. Then, sensor readings will be passed through each level of aggregators to the sink node. The studies [26–28] organize sensor nodes hierarchically and establish multi-resolution summaries of sensor data inside the network, through spatial and temporal compressions. However, such hierarchical architectures are not practical to be applied in long-thin WSNs.

### 2.5. Structure-free aggregation

The work of Fan et al. [29] considers aggregating data in a WSN without maintaining any structure. This work proposes a MAC protocol and studies the impact of randomized wait time to improve aggregation efficiency. However, this scheme may increase packet delays in long-thin WSNs.

Compared to prior aggregation/compression schemes, ALT exhibits two distinguishing features. First, while most of prior work consider selecting 'static' aggregators, ALT continuously adapts the positions of lock gates to balance between the responsiveness and the congestion of data collection. Second, while existing aggregation schemes are only to aggregate/compress the collected sensor readings, lock gates can control/adjust the amount of sensor readings generated within a cluster. By doing so, not only the amount of messages transmitted is significantly reduced but also concurrent data collections within individual clusters spatially isolated by lock gates take place.

With respect to energy conservation, many research efforts [30–34] exploit node redundancy to extend the network lifetime by selecting a subset of sensor nodes to be active while putting others to sleep to conserve energy. However, given the topology of long-thin WSNs where each sensor node usually has one potential parent node toward the sink node, such a sleep-active mechanism cannot be applied (otherwise, the network would be partitioned). In contrast, ALT strives to conserve energy by adapting the designation of lock gates to reduce the amount of messages transmitted for data collection.

## 3. PROBLEM STATEMENT

We model a long-thin WSN as a graph $\mathcal{G} = (\mathcal{V}, \mathrm{E})$, where $\mathcal{V} = \{r\} \cup \mathcal{S}$ contains the sink node $r$ and the set of sensor

nodes $\mathcal{S}$, and E contains all of the communication links. The topology of $\mathcal{G}$ may be represented as a tree rooted at sink node $r$. Each sensor node $s_j \in \mathcal{S}$ has a sensor reading generation rate $\lambda_j$, which may vary over time during the network operation. A sensor node is called a *branch node* if it has more than one child on $\mathcal{G}$. Figure 1(b) gives an example, where the branch node is marked by double circles.

We define the direction toward the sink node as the *downstream direction* and the opposite direction as the *upstream direction*. Sensor nodes are grouped into non-overlapping clusters. For each cluster, the most downstream node is designated as a *lock gate*. For convenience, we call other nodes *regular sensor nodes*. A lock gate continuously collects the sensor readings from the upstream regular sensor nodes within its cluster. Whenever a lock gate collects enough sensor readings to completely fill up one packet with payload size $L_{\max}$, the lock gate sends out the packet, termed 'collected packet', toward the sink node. Such a collected packet will not be 'collected' again by other downstream lock gates, but will be directly relayed by the downstream nodes toward the sink node. Figure 2 illustrates an example, where three clusters are formed and nodes $g_1$, $g_2$, and $g_3$ are designated as lock gates.

Given a pair of time thresholds ($T_{\min}$, $T_{\max}$), our objective is to designate lock gates (and thus adjust the corresponding clusters) such that for each lock gate $g_i$, the amount of time $T_i$ to completely fill up one collected packet of payload size $L_{\max}$ satisfies the condition of $T_{\min} \leq T_i \leq T_{\max}$. Note that the thresholds $T_{\min}$ and $T_{\max}$ are specified by the applications of the long-thin WSN to prevent sensor nodes from transmitting excessive packets and to impose an upper bound on response time, respectively. For example, for non-time-critical applications where sensor nodes are requested to frequently report their monitoring data, we set a larger $T_{\min}$ value to avoid network congestion. On the other hand, in event-driven applications, a smaller $T_{\max}$ value is set to constrain the response time.

Table I summarizes the notations used in this paper. For the ease of presentation, the sensor reading generation rate $\lambda_j$ is described as a steady-state variable, but in practice it may vary slowly during the operation of the network.

## 4. THE OPERATIONS AND ANALYSIS OF ALT

Given a long-thin WSN, ALT first randomly groups sensor nodes into several non-overlapping clusters that cover the entire network, and then designates their corresponding lock gates. Note that the sensor node closest to the sink node is always designated as a lock gate. Upon generating one sensor reading, each regular sensor node will send the reading toward its corresponding lock gate. Each lock gate $g_i$ then collects the sensor readings from the regular sensor nodes within its cluster $C(g_i)$. After collecting enough sensor readings to fill up one packet of maximum payload size $L_{\max}$, lock gate $g_i$ sends the collected packet toward the sink node. To reduce the latency of waiting to collect enough sensor readings to fill up one packet of maximum payload size $L_{\max}$, lock gate $g_i$ may dynamically adjust the size of its cluster according to the duration $T_i$ that it took to generate the previous collected packet (referring to Equation (1)). When $T_i$ is below the given lower-bound threshold $T_{\min}$, the total sensor reading generation rate within this cluster (i.e., $\sum_{s_j \in C(g_i)} \lambda_j$) has become too high, and the collected packets will be sent to the sink node more often. In this case, lock gate $g_i$ 'shrinks' its cluster by excluding certain sensor nodes to lower the total sensor reading generation rate. In contrast, when $T_i$ is above the given upper-bound threshold $T_{\max}$, the total sensor reading generation rate within this cluster becomes too low. In this case, lock gate $g_i$ 'expands' its cluster by including more sensor nodes to lower the latency of generating collected packets. Notice that within each cluster $C(g_i)$, the sensor readings sent from each regular sensor node $s_j \in C(g_i)$ may be relayed to lock gate $g_i$ in a 'pipelining' manner subject to the contention of wireless transmissions. Thus, right before lock gate $g_i$ sends out each (completely filled) collected packet toward the sink node, the percentage of sensor readings received from sensor $s_j$ within this packet is approximately equal to

$$\frac{\lambda_j}{\sum_{s_k \in C(g_i)} \lambda_k} \tag{2}$$

In other words, the amount of reported sensor readings from each sensor node is fairly proportional to the sensor reading generation rate of that sensor node.

**Table I.** Summary of notations.

| Notation | Definition |
|---|---|
| $\lambda_j$ | The sensor reading generation rate of sensor node $s_j$ |
| $C(g_i)$ | The cluster of nodes containing lock gate $g_i$ and its upstream sensor nodes |
| $L_{\max}$ | The maximum payload size of a packet |
| $\delta$ | The compression ratio ($0 < \delta \leq 1$) |
| $T_i$ | The amount of time for lock gate $g_i$ to fill up one collected packet of payload size $L_{\max}$ |
| $T_{\min}$ | The lower-bound threshold for a lock gate to shrink its cluster |
| $T_{\max}$ | The upper-bound threshold for a lock gate to expand its cluster |
| $\Delta_t$ | The waiting time for a lock gate to adjust one of its next lock gate if all of its next lock gates are busy |
| $\beta$ | A system parameter to determine whether a lock gate enters the oscillating state or not |

Before describing ALT in details, we first define the terms used in the remainder of the paper. A lock gate $g_k$ is called a *next lock gate* of lock gate $g_i$ if $g_k$ is an immediate upstream lock gate of $g_i$. In this case, $g_i$ is the *previous lock gate* of $g_k$. For instance, in Figure 2, $g_2$ is a next lock gate of $g_1$ while $g_1$ is a previous lock gate of $g_2$. Notice that each lock gate may have multiple next lock gates but has at most one previous lock gate. In addition, a lock gate is called a *leaf lock gate* if it has no next lock gate; otherwise, it is a *non-leaf lock gate*.

## 4.1. Adaptation of lock gate designation

From an initial (random) lock gate designation, lock gates execute ALT *asynchronously*, while coordinating with previous and next lock gates. Each lock gate $g_i$ measures its current $T_i$ value, 'moves' one of its next lock gates (if necessary) either downstream or upstream by one hop, and recalculates its $T_i$ value. This process is repeated until lock gate $g_i$ settles at the condition of $T_{\min} \leq T_i \leq T_{\max}$. Specifically, for each non-leaf lock gate $g_i$, two possible cases need to be addressed: $T_i < T_{\min}$ and $T_i > T_{\max}$.

### 4.1.1. Case of $T_i < T_{\min}$.

In this case, lock gate $g_i$ 'shrinks' its cluster by first querying each of its next lock gates $g_k$ for its $T_k$ value. If lock gate $g_k$ is also in the state of adjusting its own next lock gates, lock gate $g_k$ will reply to lock gate $g_i$ that itself is *busy*; otherwise, lock gate $g_k$ will reply to lock gate $g_i$ with its $T_k$ value. If lock gate $g_i$ concludes that all of its next lock gates are busy, it will wait for a $\Delta_t$ time[§] and then try again. Otherwise, lock gate $g_i$ sends a *pull* message to one next lock gate $g_k$ whose parent node, say, $s_j$ on graph $\mathcal{G}$ is not a branch node *and* whose $T_k$ value is the largest among all of lock gate $g_i$'s non-busy next lock gates. Upon receiving such a pull message, lock gate $g_k$ designates sensor node $s_j$ to become a new lock gate and ceases being a lock gate. As a result, cluster $C(g_k)$ disappears and a new cluster $C(s_j)$ emerges. For convenience, we use the term 'move' to represent such an operation. However, in the case that lock gate $g_i$ cannot find such a next lock gate (which means that the parent nodes of all of $g_i$'s non-busy next lock gates on $\mathcal{G}$ are branch nodes), the next lock gate $g_k$ that has the largest $T_k$ value is asked to move one-hop downstream. These operations are repeated until lock gate $g_i$ computes that $T_i \geq T_{\min}$.

Figure 3(a) gives an example, where $g_1$ wants to adjust one of its next lock gates $g_2$ and $g_3$. Since the parent node of lock gate $g_2$ is a branch node, lock gate $g_3$ will be asked to move downstream. When the parent nodes of all of $g_i$'s next lock gates are all branch nodes, as shown in Figure 3(b), assuming $T_2 > T_3$, lock gate $g_2$ will be asked to move to node $b$.

[§] One possibility is to set $\Delta_t = T_{\max}$ so that one of $g_i$'s next lock gates may become *non-busy*.

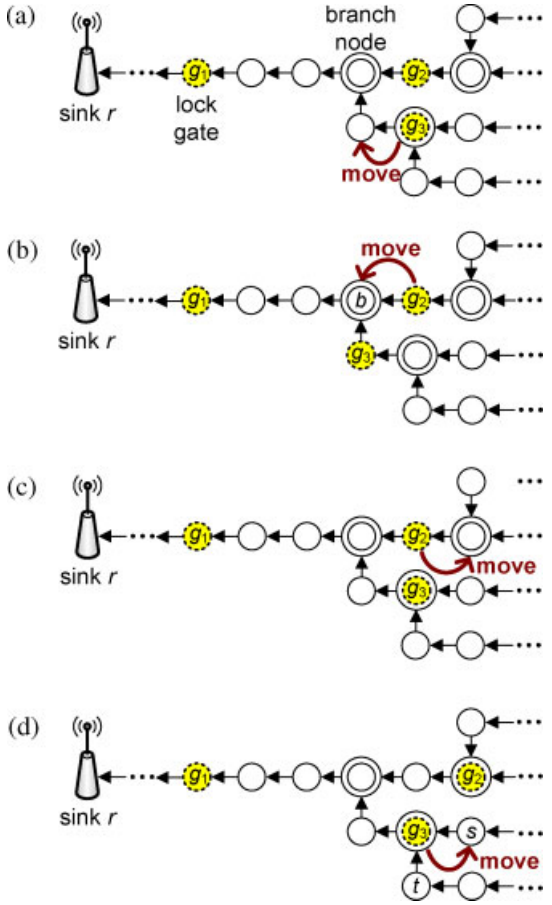



**Fig. 3.** Examples of moving next lock gates: (a) $g_1$ moves $g_3$ downstream since $T_1 < T_{\min}$, (b) $g_1$ moves $g_2$ to the branch node $b$ since $T_1 < T_{\min}$, (c) $g_1$ moves $g_2$ upstream since $T_1 > T_{\max}$, and (d) $g_1$ moves $g_3$ to node $s$ since $T_1 > T_{\max}$.

Notice that, when shrinking a cluster, ALT gives priority to move a lock gate (one-hop downstream) whose parent node on $\mathcal{G}$ is *not* a branch node. Doing so avoids excluding too many sensor nodes all at once when a lock gate is shrinking its cluster, which may *otherwise* drastically increase its $T_i$ value, and/or creating a new cluster with drastically increased cluster size (or decreased $T_i$ value) due to merging of branches. Figure 4(a) and (b) together depict one counterexample, where we assume $T_2 > T_3$. When lock gate $g_1$ simply moves one next lock gate whose $T_i$ value is the largest, lock gate $g_2$ will be moved downstream, as shown in Figure 4(b). As a result, the size of cluster $C(g_1)$ decreases drastically from 7 to 3, while the size of cluster $C(g_2)$ increases drastically from 8 to 12. In fact, ALT moves $g_3$ one-hop downstream instead.

### 4.1.2. Case of $T_i > T_{\max}$.

In this case, lock gate $g_i$ 'expands' its cluster by first querying each of its next lock gate $g_k$ for its $T_k$ value. If lock gate $g_k$ is also in the state of adjusting its own next

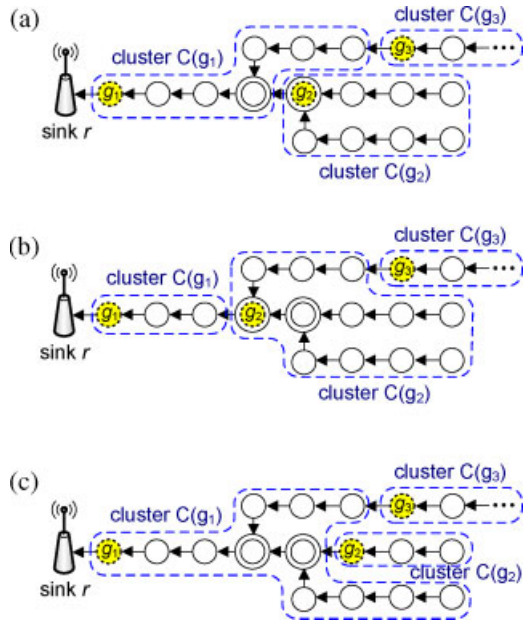



**Fig. 4.** Counterexamples of moving next lock gates: (a) the original clustering result, (b) $g_1$ simply moves $g_2$ downstream and thus drastically decreases the size of $C(g_1)$ while drastically increases the size of $C(g_2)$, and (c) $g_1$ simply moves $g_2$ upstream and thus drastically increases the size of $C(g_1)$ while drastically decreases the size of $C(g_2)$.

lock gates, lock gate $g_k$ will reply to lock gate $g_i$ that itself is busy; otherwise, lock gate $g_k$ will reply to lock gate $g_i$ with its $T_k$ value. If lock gate $g_i$ concludes that all of its next lock gates are busy, it will wait for a $\Delta_t$ time and try again. Otherwise, lock gate $g_i$ sends a *push* message to move one next lock gate $g_k$, that is not a branch node *and* has the smallest $T_k$ value, one-hop upstream. In the case that lock gate $g_i$ cannot find such a next lock gate (which means that all of $g_i$'s non-busy next lock gates are branch nodes), one next lock gate $g_k$ that has the least $T_k$ value will be moved one-hop upstream. These operations are repeated until lock gate $g_i$ computes that $T_i \leq T_{\max}$.

Figure 3(c) gives an example, where $g_1$ wants to adjust one of its next lock gates $g_2$ and $g_3$. Since lock gate $g_2$ is not a branch node, it will be moved upstream. When all of the next lock gates are branch nodes, as shown in Figure 3(d), assuming that $T_3 < T_2$, lock gate $g_3$ will be asked to move to node $s$. Notice that in the latter case, the node in which lock gate $g_3$ used to reside, node $t$, and some of node $t$'s upstream nodes will be merged into cluster $C(g_1)$.

In contrast to shrinking a cluster, ALT gives priority to move a lock gate (one-hop upstream) that is *not* a branch node when expanding a cluster. Doing so avoids including too many sensor nodes all at once when expanding a cluster, which may *otherwise* drastically decrease its $T_i$ value due to merging of branches, and/or creating a new cluster with drastically decreased cluster size (or increased $T_i$ value). Figure 4(a) and (c) together show a different counterexample with the assumption that $T_2 < T_3$. When lock gate $g_1$ simply moves its next lock gate whose $T_i$ value is the smallest, lock gate $g_2$ will be moved one-hop upstream, as shown in Figure 4(c). As a result, the size of cluster $C(g_1)$ increases drastically from 7 to 12, while the size of cluster $C(g_2)$ decreases drastically from 8 to 3. In fact, ALT moves $g_3$ one-hop upstream instead.

In ALT, sensor nodes do not need to report their sensor reading generation rates $\lambda_j$ to their corresponding lock gates. Thus, the control overhead is only incurred by transmitting query, reply, push, and pull messages between two adjacent lock gates. In subsecton 5.1, we will show that ALT's control overhead is only a small portion compared to the actual amount of data payloads.

### 4.2. Handling of leaf lock gates

For each leaf lock gate $g_l$, it will be only moved according to the push or pull requests from its previous lock gate. However, two special cases should be considered. First, when lock gate $g_l$ is asked to move upstream but itself is already a leaf node on $\mathcal{G}$, $g_l$ will simply cease being a lock gate and work as a regular sensor node. In this case, the total number of lock gates in the network decreases by one. Second, after lock gate $g_l$ has been moved downstream and computes that $T_l < T_{\min}$, lock gate $g_l$ will select one leaf node, say, $s_j$ from its cluster and designate $s_j$ as a new (leaf) lock gate. In this case, the total number of lock gates in the network increases by one.

### 4.3. Handling of oscillating lock gates

To prevent lock gates from 'oscillating' or moving back and forth between two adjacent nodes, each lock gate $g_i$ maintains a short list recording its past positions on $\mathcal{G}$. If lock gate $g_i$ finds that it has moved between two adjacent nodes (termed *oscillating nodes*) more than $\beta$ times[‖] and its previous lock gate still asks it to move to one of the oscillating nodes, lock gate $g_i$ enters the *oscillating state* and requests its previous lock gate to stop asking it to move. Lock gate $g_i$ will exit the oscillating state when either its previous lock gate asks it to move to one non-oscillating node or a pre-configured oscillating timer expires.

When each sensor node has a fixed sensor reading generation rate, the lock gates designated by ALT will eventually stabilize and converge (from an initial random designation) due to the following two factors. First, a lock gate can only move its next lock gates but cannot move its previous lock gate. In this case, clusters can stabilize *in sequence* from the downstream direction to the upstream direction. Second, ALT employs the above oscillation avoidance technique. In this case, if a lock gate finds that all of its next lock gates

[‖] The $\beta$ value will affect the convergence speed of ALT. When a fast convergence is desired, a smaller $\beta$ can be set.

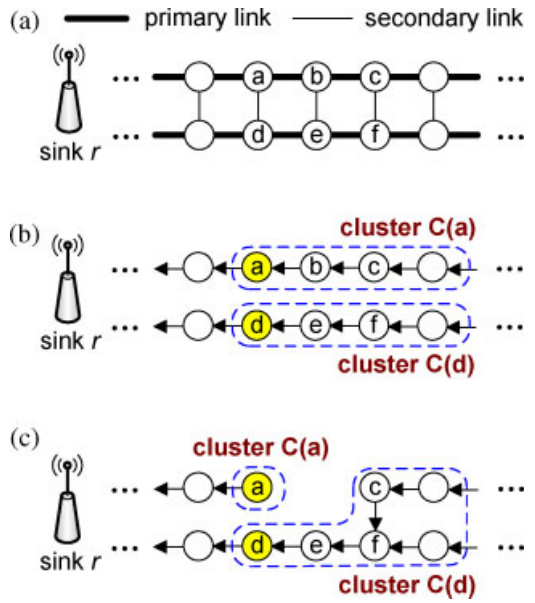



**Fig. 5.** Fault-tolerated deployment of a long-thin WSN: (a) the network topology, (b) sensor nodes transmit their sensor readings through the primary links, and (c) node *c* uses the secondary link to reach the sink node.

enter the oscillating state, the lock gate will stop moving its next lock gates (but may try later after $\Delta_t$ time).

### 4.4. Issue of fault tolerance

Till now, our discussion focuses on the assumption that each sensor node has only one potential parent node toward the sink node. For the reliability reason, a long-thin network may be deployed such that each sensor node can reach at least two downstream neighbors, as shown in Figure 5(a). In such a deployment, sensor nodes will deliver their sensor readings to the sink node through the primary links. However, when some sensor nodes fail or run out of energy, neighboring nodes can use the secondary links to forward their data. Figure 5(b) and (c) together depict an example. Initially, we have two paths $c \rightarrow b \rightarrow a \rightarrow \cdots \rightarrow r$ and $f \rightarrow e \rightarrow d \rightarrow \cdots \rightarrow r$ (formed *via* primary links), where node $r$ is the sink node. Supposing that node $b$ fails, the forwarding path from node $c$ to sink node $r$ changes from the original path $c \rightarrow b \rightarrow a \rightarrow \cdots \rightarrow r$ to the new path $c \rightarrow f \rightarrow e \rightarrow d \rightarrow \cdots \rightarrow r$. In this case, the upstream sensor nodes of node $b$ can still transmit their sensor readings to sink node $r$, even though node $b$ fails. Such a deployment ensures a higher degree of fault tolerance for long-thin WSNs.

ALT is designed to handle the following two cases:

- If a regular sensor node cannot reach its original lock gate, the sensor node will rejoin a nearest downstream cluster *via* a secondary link.
- If the new forwarding path from a regular sensor node $s_i$ to its original lock gate $g_j$ contains other lock gates, $s_i$ will leave the original cluster $C(g_j)$ and join the new cluster $C(g_k)$, where $g_k$ is $s_i$'s nearest downstream lock gate.

The first case applies when the lock gate fails or the new forwarding path cannot reach the original lock gate. The second case applies when some regular sensor nodes fail and the new forwarding path to the original lock gate passes through other lock gates. Figure 5(b) and (c) together depict an example. Initially, we have two clusters $C(a)$ and $C(d)$ with the lock gates $a$ and $d$, respectively. When node $b$ fails, the new forwarding path formed from node $c$ to sink node $r$ is no longer through lock gate $a$. This scenario fits the first case above and thus node $c$ joins the new cluster $C(d)$. From Figure 5(c), we can observe that lock gate $a$ becomes a leaf lock gate and the size of cluster $C(d)$ increases. According to the rules of ALT, lock gate $a$ may disappear and lock gate $d$ may move its next lock gates toward the downstream direction.

### 4.5. Analysis of expected extra packet latency caused by a lock gate

Clearly, a lock gate will delay the forwarding of sensor readings within its cluster toward the sink node. Below, we analyze the expected extra packet latency caused by a lock gate. We consider a cluster $C(s_1)$ that consists of a line of $K$ sensor nodes $s_1$, $s_2$, $\cdots$, and $s_K$, where node $s_1$ is the lock gate, as shown in Figure 6. To simplify the analysis, we assume that each sensor reading has the same payload size of $L$ (bytes) and the average latency to forward a sensor reading over one-hop distance is $\tau$. In addition, we assume that each sensor node $s_j$, $j = 1..K$, does not change its sensor reading generation rate (that is, $\lambda_j$ enters the steady state) during which lock gate $s_1$ waits to collect enough sensor readings to fill up one packet of maximum payload size $L_{\max}$ (i.e., $T_i$).

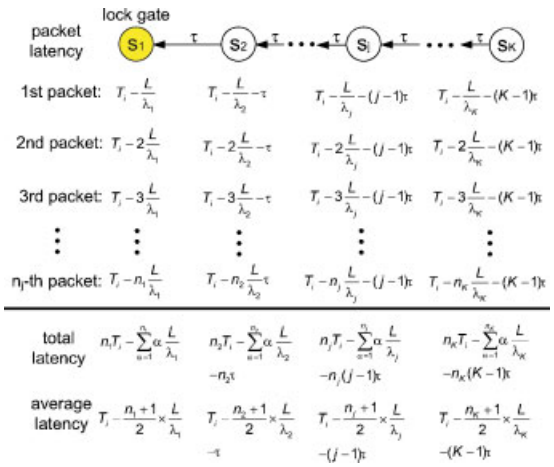



**Fig. 6.** Analysis of extra packet latency by a lock gate.

Let $n_j$ be the number of sensor readings generated by sensor node $s_j$, $j = 1 \ldots K$, during $T_i$. Notice that since $\lambda_j$ does not change during $T_i$, we have $\lambda_j = n_j / T_i$. For node $s_1$, its first packet is generated after time $L/\lambda_1$. Since node $s_1$ is the lock gate, this first packet has to wait for a duration of $T_i - L/\lambda_1$ before $s_1$ sends out a collected packet. Similarly, the second packet of node $s_1$ has to wait for a duration of $T_i - 2L/\lambda_1$, and the last packet of node $s_1$ has to wait for a duration of $T_i - n_1 L/\lambda_1$. In this case, the average extra packet latency of node $s_1$ is

$$\frac{1}{n_1}\left((T_i - \frac{L}{\lambda_1}) + (T_i - 2\frac{L}{\lambda_1}) + \cdots + (T_i - n_1\frac{L}{\lambda_1})\right)$$
$$= \frac{1}{n_1}\left(n_1 T_i - \sum_{\alpha=1}^{n_1} \alpha \frac{L}{\lambda_1}\right) = T_i - \frac{n_1+1}{2} \times \frac{L}{\lambda_1}$$

For node $s_2$, since it takes time $\tau$ to send a packet to node $s_1$, node $s_2$'s first packet has to wait a duration of $T_i - (L/\lambda_2) - \tau$. Similarly, the second packet and the last packet of node $s_2$ have to wait durations of $T_i - 2(L/\lambda_2) - \tau$ and $T_i - n_2(L/\lambda_2) - \tau$, respectively. Thus, the average extra packet latency of node $s_2$ is

$$T_i - \frac{n_2+1}{2} \times \frac{L}{\lambda_2} - \tau$$

Similarly, the average extra packet latency of node $s_j$ is

$$T_i - \frac{n_j+1}{2} \times \frac{L}{\lambda_j} - (j-1)\tau$$

Figure 6 shows the extra packet latency of each sensor node. Therefore, the average extra packet latency of all sensor nodes within cluster $C(s_1)$ incurred by lock gate $s_1$ is

$$\frac{1}{K}\left((T_i - \frac{n_1+1}{2} \times \frac{L}{\lambda_1}) + (T_i - \frac{n_2+1}{2} \times \frac{L}{\lambda_2} - \tau) + \cdots\right.$$
$$\left. + (T_i - \frac{n_K+1}{2} \times \frac{L}{\lambda_K} - (K-1)\tau)\right)$$
$$= \frac{1}{K}\left(KT_i - \frac{L}{2}\sum_{\alpha=1}^{K}\frac{n_\alpha+1}{\lambda_\alpha} - \sum_{\alpha=1}^{K-1}\alpha\tau\right)$$
$$= T_i - \frac{L}{2K}\sum_{\alpha=1}^{K}\frac{n_\alpha+1}{\lambda_\alpha} - \frac{K-1}{2}\tau \quad (3)$$

According to Equation (1), we can derive that

$$T_i = \frac{L_{\max}}{\delta \times \sum_{\alpha=1}^{K}\lambda_\alpha} \quad (4)$$

Let $n_{\text{total}}$ be the total number of packets 'generated' by all sensor nodes during $T_i$, so we have $n_{\text{total}} = \sum_{j=1}^{K} n_j$. Since each packet has a payload size of $L$, we can obtain that

$$(n_{\text{total}} \times L) \times \delta = L_{\max} \Rightarrow n_{\text{total}} = \left\lceil \frac{L_{\max}}{L \times \delta} \right\rceil$$

From Equation (2), each sensor node $s_j$, $j = 1 \ldots K$, will generate $n_j$ packets during $T_i$:

$$n_j = \frac{\lambda_j}{\sum_{\alpha=1}^{K}\lambda_\alpha} \times n_{\text{total}} = \left\lceil \frac{\lambda_j}{\sum_{\alpha=1}^{K}\lambda_\alpha} \times \frac{L_{\max}}{L \times \delta} \right\rceil \quad (5)$$

Therefore, the average extra packet latency of all sensor nodes within a cluster can be calculated by substituting Equations (4) and (5) into Equation (3).

## 5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we describe our prototyping efforts and discuss the experimental results. We use one hundred sensor nodes and one sink node. Figure 7 pictures one deployment scenario of our prototype. Each sensor node is equipped with a Jennic JN5139 chip [7] containing a micro-controller and an IEEE 802.15.4 transceiver. The transmission power of each sensor node has been adjusted to have a communication distance of approximately 30 cm. We place two adjacent nodes with a distance of 15 cm so that network connectivity can be guaranteed. An application-layer protocol has also been implemented to enforce packet reception from adjacent neighbors only. Three long-thin topologies are deployed in our experiments. The *balanced* topology (referring to Figure 8(a)) has two branches, where each branch contains 33 sensor nodes. The *unbalanced* topology (referring to Figure 8(b)) has two branches, where the long branch contains 51 sensor nodes and the short branch contains 15 sensor nodes. The *cross* topology (referring to Figure 8(c)) has three branches, where each branch contains 22 sensor nodes. We compare the performance of ALT against the *brute force* (BF) and the *fixed lock gate*

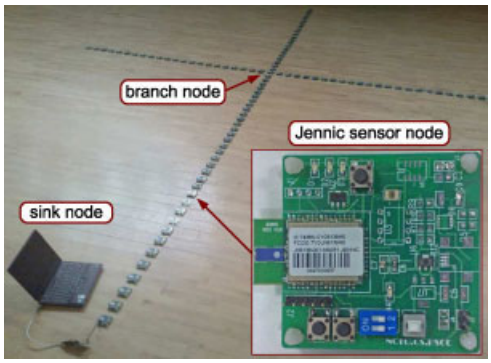



**Fig. 7.** The prototype of our long-thin WSN experiments.

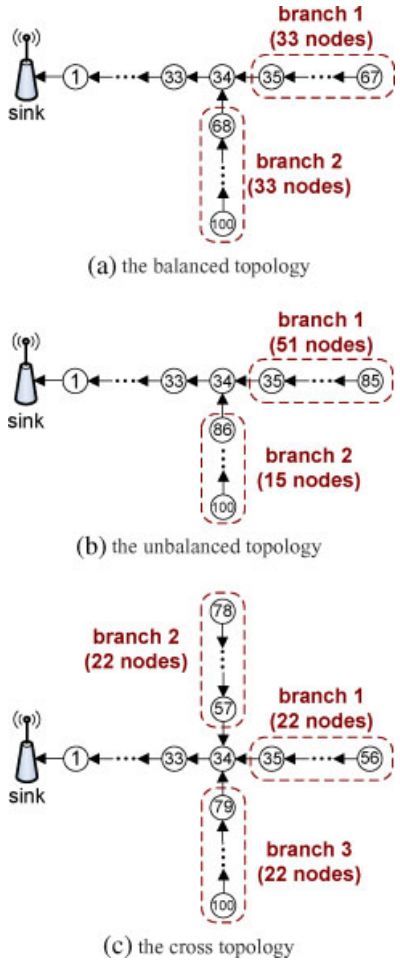



**Fig. 8.** Three long-thin topologies in our experiments.

*selection* (FLS) schemes using these three network topologies. BF does not apply any collection mechanism, so each sensor node simply relays the sensor readings from its upstream nodes to the sink node. Using FLS, each branch node is designated as a lock gate and we do not adjust the designation of lock gates during the experiments.

The size of a packet carrying one sensor reading is 15 bytes, which consists of a header of 12 bytes (according to the IEEE 802.15.4 standard [35]) and a sensor reading payload of 3 bytes. Each sensor node reports its sensor reading every $\Delta_s$ seconds, where $\Delta_s$ is randomly selected from $[(1-\gamma) \times 10, (1+\gamma) \times 10]$, and it may be changed every 30 s. Notice that when $\gamma = 0$, the sensor reading generation rate (i.e., $\lambda_i$) is 0.3 bytes/second. For each lock gate, we adopt a simple collection scheme by removing the packet headers of the received sensor reading packets and then concatenating their payloads into one single packet. In this way, we have $\delta = 1$. Since the maximum payload size of a packet defined in the IEEE 802.15.4 standard is 118 bytes, each lock gate can collect at most 39 sensor readings, so we have $L_{\max} = 39 \times 3 = 117$ bytes. The total experiment time is 10 min. In the experiments of running ALT, the measurement of messages sent by sensor nodes includes all of the control messages (e.g., query, reply, push, and pull) used to adjust the designation of lock gates. Other parameters used in the experiments are set as follows: $\beta = 3$, $\Delta_t = 2$ s, $T_{\min} = 24$ s, and $T_{\max} = 26$ s.

**Table II.** Comparison on the total amount of messages (in bytes) sent by sensor nodes in different network topologies.

| $\gamma$ value | Scheme | Balanced | Unbalanced | Cross |
|---|---|---|---|---|
| $\gamma = 0$ | ALT | 498530 | 581550 | 468982 |
| | FLS | 668962 | 785707 | 572769 |
| | BF | 928926 | 1022398 | 864898 |
| $\gamma = 0.3$ | ALT | 525009 | 606598 | 502592 |
| | FLS | 680349 | 801486 | 596287 |
| | BF | 943384 | 1096538 | 871232 |

**Table III.** Comparison on the total number of packets sent by sensor nodes in different network topologies.

| $\gamma$ value | Scheme | Balanced | Unbalanced | Cross |
|---|---|---|---|---|
| $\gamma = 0$ | ALT | 33865 | 35517 | 32225 |
| | FLS | 99915 | 128396 | 78767 |
| | BF | 238899 | 267690 | 217392 |
| $\gamma = 0.3$ | ALT | 35878 | 35788 | 33033 |
| | FLS | 103750 | 128563 | 81621 |
| | BF | 240855 | 282288 | 218911 |

## 5.1. Communication costs

We use the amount of messages and the number of packets successfully forwarded to the sink node to measure communication cost of data collection. Table II lists the total amount of messages (in bytes) successfully forwarded[¶] to the sink node by sensor nodes in different network topologies. Without using any collection scheme, BF exhibits the highest amount of messages. By dynamically adjusting the positions of lock gates according to the network condition, ALT enjoys a lower amount of messages compared with FLS. It can be observed that when $\gamma = 0$, ALT saves 18.1–26.0% and 43.1–46.3% of the amount of messages compared with FLS and BF, respectively. When $\gamma = 0.3$, ALT saves 15.7–24.3% and 42.3–44.7% of the amount of messages compared with FLS and BF, respectively. These results show the effectiveness of ALT. Notice that the three schemes all suffer from the highest amount of messages under the unbalanced topology, because this topology has the longest branch (with 51 sensor nodes).

Table III lists the total number of packets successfully forwarded to the sink node by sensor nodes in different network topologies. Using BF, sensor nodes forward the

[¶] Due to wireless contention and impairment, packet losses and retransmissions do occur, which are evaluated in subsection 5.3.

most number of packets, because they simply relay sensor readings to the sink node. ALT incurs the smallest number of packets among all three schemes because it adaptively clusters sensor nodes *via* lock gates and collects their packets accordingly. It can be observed that when $\gamma = 0$, ALT saves 59.1–72.3% and 85.2–86.7% of the number of packets compared with FLS and BF, respectively. When $\gamma = 0.3$, ALT saves 59.5–72.2% and 84.9–87.3% of the number of packets compared with FLS and BF, respectively. These results demonstrate that ALT significantly reduces the number of packets forwarded by sensor nodes, which can greatly alleviate network congestion and conserve energy.

Notice that our measurement includes all the control messages. From Tables II and III, we can observe that ALT incurs very low traffic even when control overheads are included. Thus, the impact of control overheads caused by ALT's adaption of lock gates is light-weight.

## 5.2. Adaptive designation of lock gates

To demonstrate the adaptability of ALT to varying sensor reading generation rates, we deploy a sink node (of ID 0) and a line of 50 sensor nodes (of IDs 1 to 50), where the node with ID 1 is the most downstream sensor node. The duration of the experiment is 126 min and we monitor the (changing) number of lock gates and their designation (or positions) over time. All of the sensor nodes have the same sensor reading generation rate (λ), which changes every 3 min as shown in Figure 9(a). For instance, starting at 0.2 bytes/second, λ remains at the same rate until the 36th minute, increases to 0.6 bytes/second at the 66th minute, remains at the same rate until the 81st minute, and then decreases. Figure 9(b) depicts the changing designation of lock gates, as dots, over time. For instance, at the 0th minute, there are six lock gates randomly designated at nodes of IDs 1, 4, 18, 20, 25, and 48. Before the 36th minute, λ is not changed and thus the positions of lock gates stabilize at nodes of IDs 1, 16, 31, and 45 at the 24th minute. When λ increases, the size of the clusters decreases and thus the number of lock gates increases accordingly. Between the 66th and the 81st minutes, λ remains stable and thus the designation of lock gates are only slightly adjusted. For instance, at the 72nd minute, eight lock gates are designated at nodes of IDs 1, 6, 12, 18, 25, 32, 38, and 47. After the 81st minute, when λ decreases, the number of lock gates also decreases and the size of clusters increases. After the 117th minute, the designation of lock gates remains stable because λ does not change. Since all of the sensor nodes have the same λ value, we also observe that the distance between any two adjacent lock gates is quite similar at most time instances. Such a phenomenon is more visible when the number of lock gates is smaller. These observations demonstrate that ALT can efficiently adjust the size of each cluster (and designate the lock gate accordingly) based on the traffic sent from the sensor nodes in that cluster.

Using the same network topology in the previous experiment, we also demonstrate the adaptability of ALT when

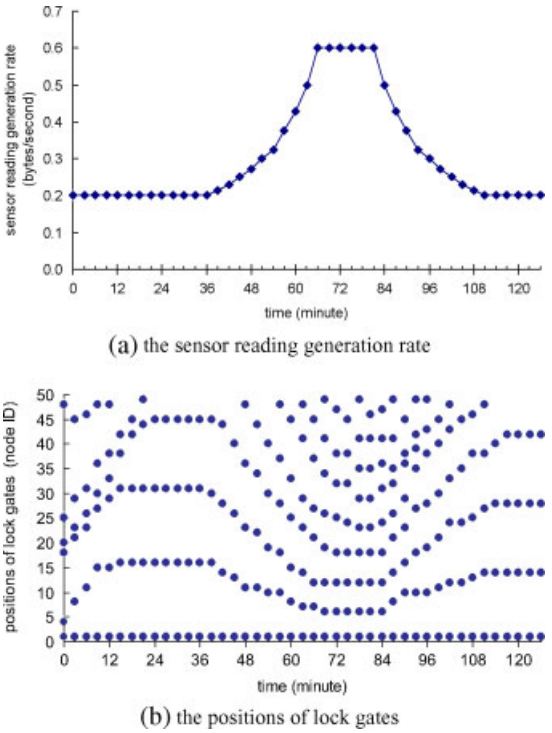



**Fig. 9.** The change of the positions of lock gates when all sensor nodes have the same sensor reading generation rate.

sensor nodes have different sensor reading generation rates (λ). Specifically, the λ value of sensor nodes of IDs 1 to 25 increases while that of sensor nodes of IDs 26 to 50 decreases over time, as shown in Figure 10(a). For convenience, we use the terms 'downstream part' and 'upstream part' to represent the sensor nodes of IDs 1 to 25 and of IDs 26 to 50, respectively. The duration of the experiment is 120 min. Beginning with the same random lock gate designation as the previous experiment, Figure 10(b) shows the changes of lock gate designation over time. We observe that before the 60th minute, most lock gates are located at the upstream part because sensor nodes in the upstream part have a higher sensor reading generation rate. Thus, ALT shrinks the sizes of clusters in the upstream part and thus designates more lock gates. After the two sensor reading generation rates cross around the 66th minute, the behavior reverses itself such that most lock gates move to the downstream part because sensor nodes in the downstream part have a higher sensor reading generation rate.

## 5.3. Impact on MAC-layer behaviors

We also evaluate the impact of lock gates on the MAC-layer behaviors in terms of the number of packet retransmissions and the number of packet losses by reusing the three network topologies in Figure 8. We assign sensor nodes with IDs 1

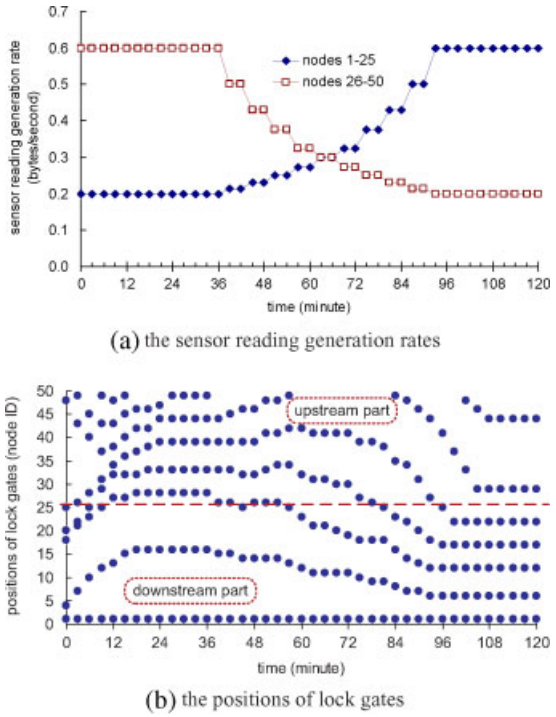



**Fig. 10.** The change of the positions of lock gates when sensor nodes have different sensor reading generation rates.

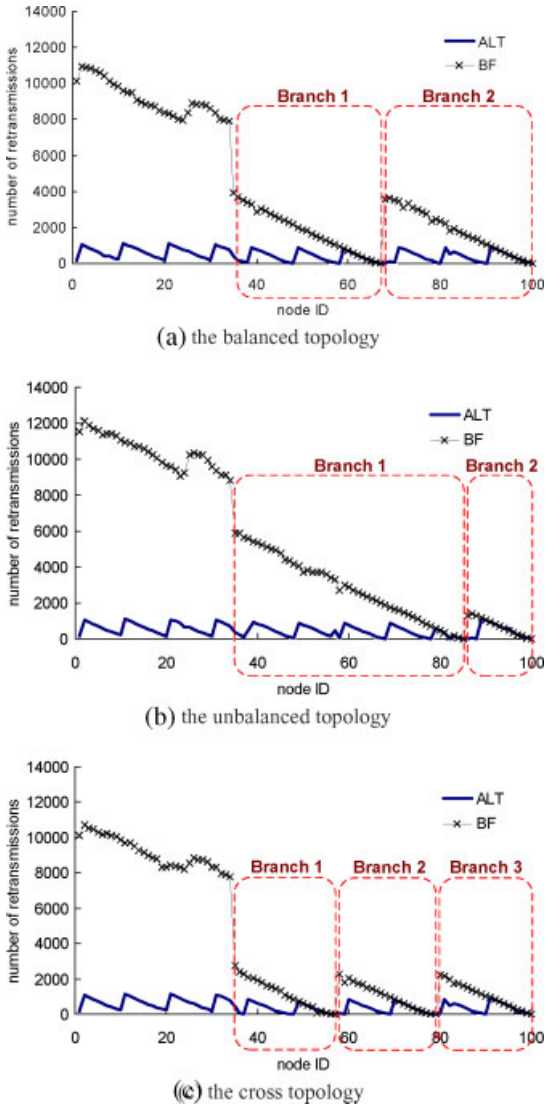



**Fig. 11.** Comparison on the numbers of packet retransmissions under ALT and BF.

to 100, where the node with ID 1 is the most downstream sensor node. In the balanced topology, ten sensor nodes with IDs 1, 10, 20, 30, 38, 48, 58, 70, 80, and 90 are designated as lock gates. In the unbalanced topology, eleven sensor nodes with IDs 1, 10, 20, 30, 38, 48, 58, 68, 78, 88, and 98 are designated as lock gates. In the cross topology, ten sensor nodes with IDs 1, 10, 20, 30, 37, 47, 59, 69, 80, and 90 are designated as lock gates. Each sensor node reports its sensor reading every 5 s, so the sensor reading generation rate $\lambda_i$ is 0.6 bytes/second. The duration of each experiment is 10 min.

Figure 11 compares the number of MAC-layer retransmissions incurred by ALT and BF in the three long-thin topologies. Without lock gates, we observe that the length of a branch directly affects the number of sensor readings generated, and hence negatively affects the number of packet retransmissions due to contention, such that the unbalanced topology (containing the longest branch) incurs more retransmissions. The branch node with ID 34 suffers a steep jump of the number of retransmissions in comparison to its neighboring upstream nodes due to the heavy contention caused by the traffic coming from upstream nodes. In contrast, using ALT, lock gates dynamically adjust the cluster sizes according to the traffic loads, so that the number of retransmissions is not only reduced but also not affected by the topology. In Figure 11, the numbers of retransmissions incurred by ALT in all three topologies exhibit very similar saw-tooth curves with valleys occurring at the lock gate nodes due to the fact that lock gates collect the packets within the corresponding clusters and reduce the number of transmissions.

We now compare the number of packet losses at each sensor node in the three network topologies. As shown in Figure 12, using BF, sensor nodes closer to the sink node incur a higher number of packet losses because these sensor nodes have to relay more sensor readings from upstream nodes. In particular, the unbalanced topology suffers more packet losses than the other two topologies due to the existence of the longest branch and hence the highest contention. The branch node with ID 34 incurs more packet losses than its upstream neighboring nodes due to higher contention. In contrast, using ALT, the numbers of packet losses are quiet small in all three topologies, which indicates that lock gates significantly reduce the number of transmissions and mitigate the contention.

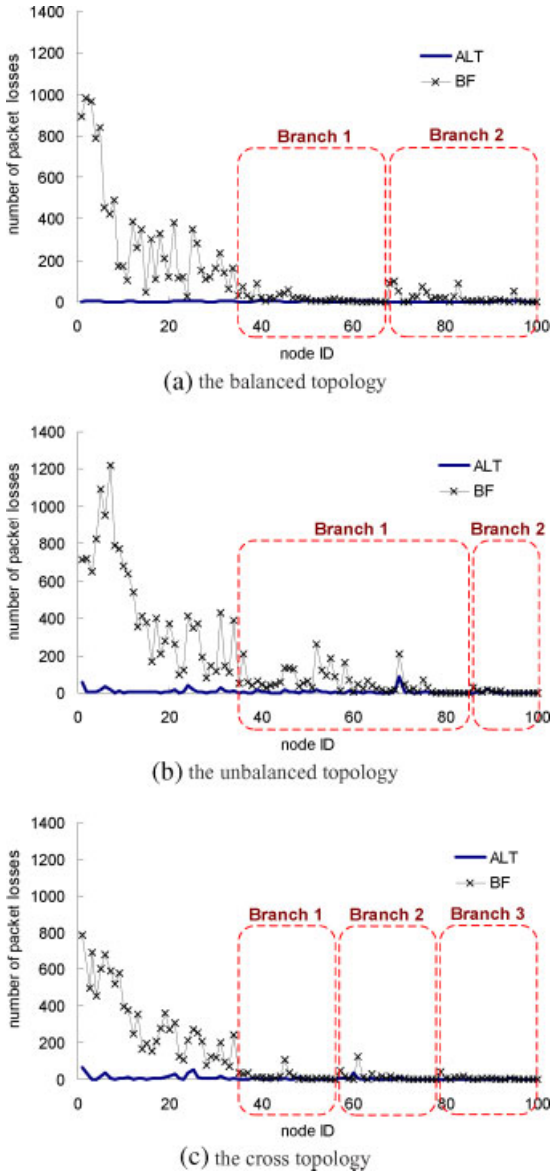



**Fig. 12.** Comparison on the numbers of packet losses under ALT and BF.

## 5.4. Average packet latency

In the same experiments described in subsection 5.3, we also measure the average latency per successfully delivered packet from each sensor node to the sink node, as shown in Figure 13. Using BF, sensor nodes simply relay the sensor readings toward the sink node, and the average latency is directly influenced by how far a sensor node is located away from the sink node. As we can observe, nodes in the unbalanced topology suffer from higher latency due to the longest branch. Using ALT, since the lock gates have to keep the received sensor readings locally while waiting to receive enough number of sensor readings to fill up a packet of maximum payload size, the packet latency is higher than using BF. According to the analysis in subsection 4.5, we calculate the difference between the average packet latency using ALT and that using BF. Since the cluster size in these experiments is about ten, we have $K = 10$. In addition, the payload size of a sensor reading is 3 bytes and each sensor node generates one sensor reading every 5 s, so we have $L = 3$ and $\lambda_j = 0.6$ for each sensor node $s_j$, $j = 1 \ldots K$. By Equation (4), we can derive that

$$T_i = \frac{L_{\max}}{\delta \times \sum_{\alpha=1}^{K} \lambda_\alpha} = \frac{117}{1 \times 10 \times 0.6} = 19.5$$

Using Equation (5), for each $j = 1 \ldots K$, we can obtain that

$$n_j = \left\lceil \frac{\lambda_j}{\sum_{\alpha=1}^{K} \lambda_\alpha} \times \frac{L_{\max}}{L \times \delta} \right\rceil = \left\lceil \frac{0.6}{10 \times 0.6} \times \frac{117}{3 \times 1} \right\rceil = 4$$

The value of $\tau$ can be measured by the average packet latency using BF. Thus, from Figure 13, we can obtain that $\tau \approx 0.29$. Therefore, according to Equation (3), we calculate the expected extra packet latency using ALT by

$$\begin{aligned} & T_i - \frac{L}{2K} \sum_{\alpha=1}^{K} \frac{n_\alpha + 1}{\lambda_\alpha} - \frac{K-1}{2} \tau \\ &= 19.5 - \frac{3}{2 \times 10} \times (10 \times \frac{4+1}{0.6}) - (\frac{10-1}{2} \times 0.29) \\ &= 5.695 \end{aligned}$$

Table IV gives the average packet latency of each sensor node using ALT and BF (which are derived from Figure 13). It can be observed that the difference between the average packet latency using ALT and that using BF is 5.17, which is close to the analyzed result (i.e., 5.695). The slight inaccuracy is due to the fact that not all clusters have ten sensor nodes in our experiments. The above observation validates the soundness of our analysis in subsection 4.5.

From Figure 13, it can be observed the average packet latency using ALT exhibits saw-tooth curves in all three topologies due to the fact that every lock gate transmits a collected packet after $T_i$ seconds as calculated by Equation (1) while sensor readings arrive at the corresponding lock gates at different moments in time. Although BF incurs low latency, BF suffers from large numbers of retransmissions and packet losses. It is clear to conclude that ALT balances between the latency and the congestion in long-thin WSNs.

**Table IV.** Average packet latency of each sensor node using ALT and BF (unit: seconds).

| Scheme | Balanced | Unbalanced | Cross | Average |
|---|---|---|---|---|
| ALT | 5.58 | 5.26 | 5.54 | 5.46 |
| BF | 0.29 | 0.29 | 0.28 | 0.29 |
| Difference | 5.29 | 4.97 | 5.26 | 5.17 |

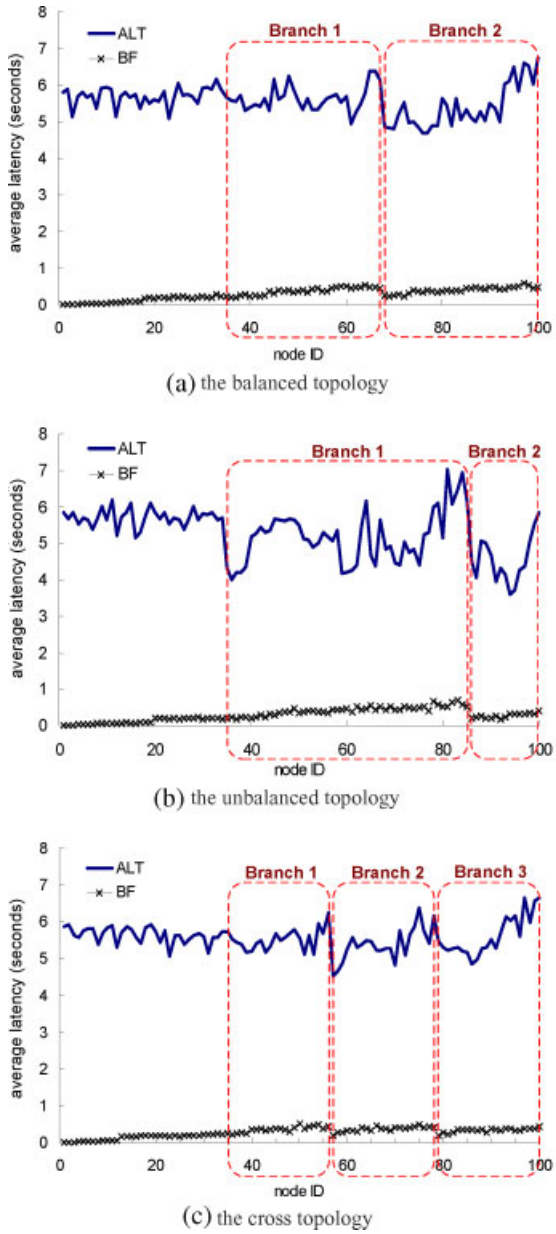



**Fig. 13.** Comparison on the average packet latency under ALT and BF.

## 6. CONCLUSIONS

Many realistic WSN applications dictate the deployment of long-thin topology which demands new data collection schemes. This paper describes the ALT lock gate designation scheme which, (1) designates multiple lock gates within a long-thin WSN to regulate data collection and (2) adapts the designation of lock gates dynamically in response to changing sensor reading generation rates of sensor nodes. ALT balances between the responsiveness and the congestion of data collection, and mitigates the funneling effect by regulating (collected) data that could be transmitted downstream and spatially separating areas where packets are transmitted. Using the Jennic JN5139 wireless micro-controllers, we evaluate the performance of ALT *via* experiments with prototyped long-thin WSNs. Experimental results demonstrate the merits of ALT and reveal the impact of lock gate designation on MAC-layer behaviors and latency.

## REFERENCES

1. Szewczyk R, Osterweil E, Polastre J, Hamilton M, Mainwaring A, Estrin D. Habitat monitoring with ensor networks. *Communications of the ACM* 2004; **47**(6): 34--40.
2. Helal S, Mann W, El-Zabadani H, King J, Kaddoura Y, Jansen E. The gator tech smart house: a programmable pervasive space. *IEEE Computer* 2005; **38**(3): 50–60.
3. Tseng YC, Wang YC, Cheng KY, Hsieh YY. iMouse: an integrated mobile surveillance and wireless sensor system. *IEEE Computer* 2007; **40**(6): 60–66.
4. Pan MS, Fang HW, Liu YC, Tseng YC. Address assignment and routing schemes for ZigBee-based long-thin wireless sensor networks. In *IEEE Vehicular Technology Conference*, May 2008; 173–177.
5. Ahn GS, Hong SG, Miluzzo E, Campbell AT, Cuomo F. Funneling-MAC: a localized, sink-oriented MAC for boosting fidelity in sensor networks. In ACM International Conference on Embedded Networked Sensor Systems, October 2006; 293–306.
6. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. *IEEE Communications Magazine* 2002; **40**(8): 102–114.
7. Jennic wireless microcontrollers. [Online]. Available at: www.jennic.com
8. De D. A distributed algorithm for localization error detection-correction, use in in-network faulty reading detection: applicability in long-thin wireless sensor networks. In *IEEE Wireless Communications and Networking Conference*, April 2009.
9. Hua C, Yum TSP. Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks. *IEEE/ACM Transactions on Networking* 2008; **16**(4): 892–903.
10. Kalpakis K, Dasgupta K, Namjoshi P. Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks. *Computer Networks* 2003; **42**(6): 697–716.
11. Krishnamachari B, Estrin D, Wicker S. The impact of data aggregation in wireless sensor networks. In *IEEE International Conference on Distributed Computing Systems Workshops*, July 2002; 575–578.
12. Intanagonwiwat C, Govindan R, Estrin D, Heidemann J, Silva F. Directed diffusion for wireless

sensor networking' *IEEE/ACM Transactions on Networking* 2003; **11**(1): 2–16.
13. Madden S, Franklin MJ, Hellerstein JM, Hong W. TAG: a tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review* 2002; **36**: 131--146.
14. Harris III AF, Kravets R, Gupta I. Building trees based on aggregation efficiency in sensor networks. *ACM Ad Hoc Networks* 2007; **5**(8): 1317–1328.
15. von Rickenbach P, Wattenhofer R. Gathering correlated data in sensor networks. In *ACM Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, October 2004; 60–66.
16. Pattem S, Krishnamachari B, Govindan R. The impact of spatial correlation on routing with compression in wireless sensor networks. *ACM Transactions on Sensor Networks* 2008; **4**(4): 1–33.
17. Younis O, Krunz M, Ramasubramanian S. Node clustering in wireless sensor networks: recent developments and deployment challenges. *IEEE Network* 2006; **20**(3): 20–25.
18. Basagni S. Distributed clustering for *ad hoc* networks. In *IEEE International Symposium on Parallel Architectures, Algorithms and Networks*, June 1999; 310–315.
19. Kuhn F, Moscibroda T, Wattenhofer R. Initializing newly deployed *ad hoc* and sensor networks. In *ACM International Conference on Mobile Computing and Networking*, September 2004; 260–274.
20. Heinzelman WB, Chandrakasan AP, Balakrishnan H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications* 2002; **1**(4): 660–670.
21. Younis O, Fahmy S. HEED: a hybrid, energy-efficient, distributed clustering approach for *ad hoc* sensor networks. *IEEE Transactions on Mobile Computing* 2004; **3**(4): 366–379.
22. Zhu Y, Vedantham R, Park SJ, Sivakumar R. A scalable correlation aware aggregation strategy for wireless sensor networks. In *IEEE International Conference on Wireless Internet*, July 2005; 122–129.
23. Lindsey S, Raghavendra C, Sivalingam KM. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems* 2002; **13**(9): 924–935.
24. Du K, Wu J, Zhou D. Chain-based protocols for data broadcasting and gathering in sensor networks. In *IEEE International Symposium on Parallel and Distributed Processing*, April 2003.
25. Chen YP, Liestman AL, Liu J. A hierarchical energy-efficient framework for data aggregation in wireless sensor networks. *IEEE Transactions on Vehicular Technology* 2006; **55**(3): 789–796.
26. Ganesan D, Estrin D, Heidemann J. Dimensions: why do we need a new data handling architecture for sensor networks? *ACM SIGCOMM Computer Communication Review* 2003; **33**(1): 143–148.
27. Ganesan D, Greenstein B, Estrin D, Heidemann J, Govindan R. Multiresolution storage and search in sensor networks. *ACM Transactions on Storage* 2005; **1**(3): 277–315.
28. Wang YC, Hsieh YY, Tseng YC. Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications. *IEEE Transactions on Computers* 2009; **58**(6): 827–838.
29. Fan KW, Liu S, Sinha P. Structure-free data aggregation in sensor networks. *IEEE Transactions on Mobile Computing* 2007; **6**(8): 929–942.
30. Chen B, Jamieson K, Balakrishnan H, Morris R. Span: an energy-efficient coordination algorithm for topology maintenance in *ad hoc* wireless networks. *ACM Wireless Networks* 2002; **8**(5): 481–494.
31. Cardei M, Du DZ. Improving wireless sensor network lifetime through power aware organization. *ACM Wireless Networks* 2005; **11**(3): 333–340.
32. Zou Y, Chakrabarty K. A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. *IEEE Transactions on Computers* 2005; **54**(8): 978–991.
33. Xing G, Wang X, Zhang Y, Lu C, Pless R, Gill C. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks* 2005; **1**(1): 36–72.
34. Zhao Q, Gurusamy M. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions on Networking* 2008; **16**(6): 1378–1391.
35. LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Std 802.15.4-2003, Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), *IEEE*, 2003.

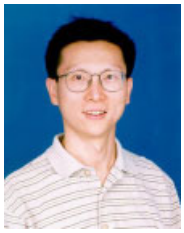
## Authors' Biographies

**You-Chiun Wang** received his B.Eng and M.Eng degrees in Computer Science and Information Engineering from the National Chung-Cheng University and the National Chiao-Tung University, Taiwan, in 2001 and 2003, respectively. He obtained his Ph.D. degree in Computer Science from the National Chiao-Tung University, Taiwan, in October of 2006. Currently, he is a postdoctoral research fellow at the Department of Computer Science,

National Chiao-Tung University, Taiwan. His research interests include wireless network and mobile computing, communication protocols, and wireless sensor networks. Dr. Wang served as a guest editor of The Computer Journal (2009), on the editorial broad of IARAI International Journal on Advances in Networks and Services (2009-present), and as TPC members of several conferences. He is a member of the IEEE.

**Che-His Chuang** received his B.Eng and M.Eng degrees in Computer Science from the National Chiao-Tung University, Taiwan, in 2007 and 2009, respectively. His research interests include wireless communication and sensor networks.

**Yu-Chee Tseng** obtained his Ph.D. in Computer and Information Science from the Ohio State University in January of 1994. He is Professor (2000-present), Chairman (2005-2009), and Associate Dean (2007-present) at the Department of Computer Science, National Chiao-Tung University, Taiwan. He is also Adjunct Chair Professor at the Chung-Yuan Christian University (2006-present). Dr. Tseng received Outstanding Research Award, by National Science Council, ROC, twice in periods 2001-2002 and 2003-2005, Best Paper Award (International Conference on Parallel Processing, 2003), the Elite I.T. Award in 2004, and the Distinguished Alumnus Award, by the Ohio State University, in 2005. His research interests include mobile computing, wireless communication, and parallel and distributed computing. Dr. Tseng serves on the editorial boards for Telecommunication Systems (2005-present), IEEE Transactions on Vehicular Technology (2005-2009), IEEE Transactions on Mobile Computing (2006-present), and IEEE Transactions on Parallel and Distributed Systems (2008-present).



**Chien-Chung Shen** received his B.S. and M.S. degrees from National Chiao-Tung University, Taiwan, and his Ph.D. degree from UCLA, all in computer science. He was a senior research scientist at Bellcore (now Telcordia) Applied Research working on control and management of broadband networks. He is now an associate professor in the Department of Computer and Information Sciences of the University of Delaware. His research interests include ad hoc and sensor networks, dynamic spectrum management, control and management of broadband networks, distributed object and peer-to-peer computing, and simulation. He is a recipient of NSF CAREER Award, and his research has been sponsored by NSF, NASA, Army Research Lab, RAND, and industrial companies. He is a member of both ACM and IEEE.


DOI: 10.1002/wcm