WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2010; **10**:1078–1101
Published online 15 July 2009 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/wcm.824

A frequency-aware data-centric mechanism for wireless sensor networks

Chih-Yung Chang[1]*, Jang-Ping Sheu[2], Sheng-Wen Chang[1] and Yu-Chieh Chen[1]

[1]*Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taipei, Taiwan*
[2]*Department of Computer Science and Information Engineering, National Tsing Hua University, Hsinchu, Taiwan*

Summary

Wireless sensor networks (WSNs) are characterized by their low bandwidth, limited energy, and largely distributed deployment. To reduce the flooding overhead raised by transmitting query and data information, several data-centric storage (DCS) mechanisms are proposed. However, the locations of these data-centric nodes significantly impact the power consumption and efficiency for information queries and storage capabilities, especially in a multi-sink environment. This paper proposes a novel dissemination approach, which is namely the dynamic data-centric routing and storage mechanism (DDCRS), to dynamically determine locations of data-centric nodes according to sink nodes' location and data collecting rate and automatically construct shared paths from data-centric nodes to multiple sinks. To save the power consumption, the data-centric node is changed when new sink nodes participate when the WSNs or some queries change their frequencies. The simulation results reveal that the proposed protocol outperforms existing protocols in terms of power conservation and power balancing. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: data query; data-centric storage; data dissemination; wireless sensor networks; path sharing

1. Introduction

A wireless sensor network (WSN) is composed of a few sink nodes and an extremely large number of sensor nodes that are densely deployed in a particular area. A sink node is a control center which typically initiates a request for collecting interested information. Linked by a wireless medium, the sensor nodes perform distributed sensing tasks and store particular sensing information for queries. One critical problem in sensor networks is how to effectively provide sink and sensor nodes with efficient data retrieving and storing, respectively. Previous solutions to this problem can be classified into three categories: local storage (LS), external storage (ES), and data-centric storage (DCS).

In LS mechanisms, data are stored in sensor nodes' local memory when an event is detected. Since the sink node does not know which sensor nodes store the interested data, a sink node intending to collect the interested data typically executes a blind flooding over the whole WSN to send a query packet defining the criteria of interests. ES, on the other hand, proposes

*Correspondence to: Chih-Yung Chang, Department of Computer Science and Information Engineering, Tamkang University, 151 Ying-Chuan Road, Tamsui, Taipei, Taiwan.
†E-mail: cychang@mail.tku.edu.tw

another alternative mechanism. Once a sensor node detects an event, the data are stored at the external sink. Although there is no cost for sink query, it may waste a lot of energy for transmitting data to the sink that is not interested in the data. In the DCS mechanism, there are numbers of data-centric nodes selected from the WSN that are responsible for handling data storage and retrieval. When an event is detected by a sensor node, data are stored by name at a corresponding data-centric node. Because all sensor nodes and sink nodes are aware of the information in data-centric nodes, they do not need to apply blind flooding for sending data or queries to data-centric nodes.

In literatures, previous studies [1–3] have proposed an efficient DCS mechanism for WSNs. The hash function of the geographic hash table (GHT) is used to map events to locations of data-centric nodes in the monitoring area. A sensor node uses the hash function of the GHT to obtain a location, after which the sensed information will then be stored in the data-centric node closest to the location using the GPSR [4] routing protocol. When a sink node intends to collect an event information, it uses the hash function of the GHT to obtain the location where the event is stored and then adopts a GPSR [4] routing protocol to send a query packet to the data-centric node which is the sensor node closest to the location. Upon receiving the query from the sink node, the data-centric node replies with the requested data.

A previous work, GEM [5] developed a DCS and a node-to-node routing mechanism without the need for geographic information. They constructed the virtual polar coordinate space (VPCS), where each node is given a label consisted of angle and level. Similar to the approach developed in Reference [1], a static hash table is used to convert events to their corresponding labels, and therefore packets are routed according to the relationship of labels in VPCS. In Reference [6], a robust DCS architecture is developed in a mobility environment. In this approach, a rendezvous region is used instead of a rendezvous point to increase robustness. Similar to References [1,2,5], a static hash table is applied to map the all possible events to regions of WSN. Due to the existence of several data-centric servers in a region, the proposed mechanism increases the robustness of data-centric node failure. An index-based architecture [7] has been proposed to reduce unnecessary transmission in situations where the ratio of interested information required from sink node to the sensing data is relative low. A static hash table is also used to develop a data-centric ring-based index. Q. Fang, J. Gao and L. J. Guibas [8] proposed a data-centric mechanism, based on the GLIDER [9] framework, to speed up a sink's query and reduce the control overhead for finding data-centric nodes. The WSN stores the same data in some backup nodes so that the sink nodes may access the backup nodes. This way, the query efficiency can be significantly improved. In study [10], a double rulings scheme chooses the rendezvous nodes along a continuous curve to store data instead of one or multiple sensor nodes. Therefore, the replication curve can increase the fault tolerance. Moreover, the double rulings scheme also provides distance-sensitive retrieval scheme such that the sink node sends a query to travel along a curve that intersects the replication curve as quickly as possible. When the sink node is close to the sensor node that sends the sensing data to the replication curve, it can find the data quickly.

Although the aforementioned papers devoted themselves to develop DCS architecture in different environments, most of them did not consider the multiple sinks environment and the factor of query frequencies. Using a static hash table to determine the location of a data-centric node might raise communication overhead which highly relies on the locations of the sink nodes and the frequencies of data delivery, especially in a multi-sink environment. Moreover, if the information of a specific event is stored in a fixed data-centric node for a long time, sensors nodes that are near the data-centric nodes would likely exhaust their energy due to frequent data forwarding, resulting in unbalanced power consumption among the WSNs.

This paper aims to develop path sharing and DCS mechanisms for a multi-sink environment. Firstly, a dynamic DCS mechanism is proposed to dynamically determine the location of data-centric nodes according to the location and the requested frequency of multiple sink nodes. Problems raised because of the change of data-centric node are investigated and resolved. An efficient share-path routing mechanism is also presented to construct a shared path from data-centric nodes to multiple sink nodes, reducing the redundant packet transmission and the number of forwarding nodes, and therefore saving the power consumptions. The impacts of the proposed DCS and the path sharing mechanisms are investigated. Simulation results reveal that the proposed mechanisms reduce the redundant packet transmission, both saving power and bandwidth consumption, and therefore prolonging network life.

The remainder of the paper is organized as follows. In Section 2, a multi-sink network environment is described. Issues for determining data-centric locations and shared routing are analyzed. Section 3 presents an overview of the developed mechanisms, and illustrates the data-centric routing mechanism and dynamic DCS mechanism in detail. Section 4 describes the developed protocols and discusses related issues. Performance study is presented in Section 5 and finally, Section 6 concludes the paper.

2. Environment and Problem Analysis

This section describes the network environment and analyzes the problems of route sharing and dynamic data-centric node selection.

2.1. Network Model

The network model is similar to previous works [1,2,3,5,7,8] that have developed data-centric mechanisms for WSNs. All sensor nodes are stationary and randomly deployed in the monitoring area. There are no obstacles and holes existing in the WSNs. Each sensor node is aware of its own location using GPS [11] or other techniques such as References [12,13] and exchanges location information with one-hop neighbors through beacons. The events will be randomly occurred at the monitoring region and their values vary with the time. Once a sensor node detects the event data, it adopts the pre-defined hash table [2] to obtain the location, say *l*, for storing data, and then transmits the data to the data-centric node that is closet to location *l* using the GPSR routing protocol [4]. Multiple stationary sink nodes exist in the sensor network and their locations are known by all sensor nodes. Each sink node might be interested in monitoring some event for a specific duration by returning event values from the data-centric node in a constant frequency. Therefore, the query packet contains information including event values, frequency, and duration. When a sink node intends to send a query request to the data-centric node, it uses a pre-defined hash table to obtain the data-centric location of interested events and then sends the query packet using the GPSR routing protocol. Upon receiving the query packet from a sink node, the data-centric node applies the proposed mechanism to determine a better location to play the role of data-centric node according to the locations and frequencies of those sink nodes that have sent their query packets to the data-centric node. Then the data-centric node periodically replies the interested data to the sink nodes by applying the proposed data-centric routing mechanism.

2.2. Problem Analysis

In previous works [1–3], some peer-to-peer algorithms, such as Chord [14], Pastry [15], Tapestry [16], CAN [17], were adopted to determine the locations of data-centric nodes. However, the selection of data-centric nodes without considering the locations and the requested data collection frequencies of sink nodes might raise the problem of inefficient communication. Figure 1 can be taken as an example to depict the situation. Three different sinks, *X*, *Y*, *Z*, intend to query the information of event *A* with the required reply frequency, 50, 10, and 20, respectively. Figure 1(a) depicts the communication path using algorithms proposed in previous works [14–17]. Sink *X* requires the largest reply frequency, but its location is the farthest from the data-centric node *A*. Therefore, a large communication overhead will need to be spent to transmit interested information from data-centric node *A* to sink *X*. According to the locations and requested reply frequencies of sink nodes *X*, *Y*, and *Z*, the proposed

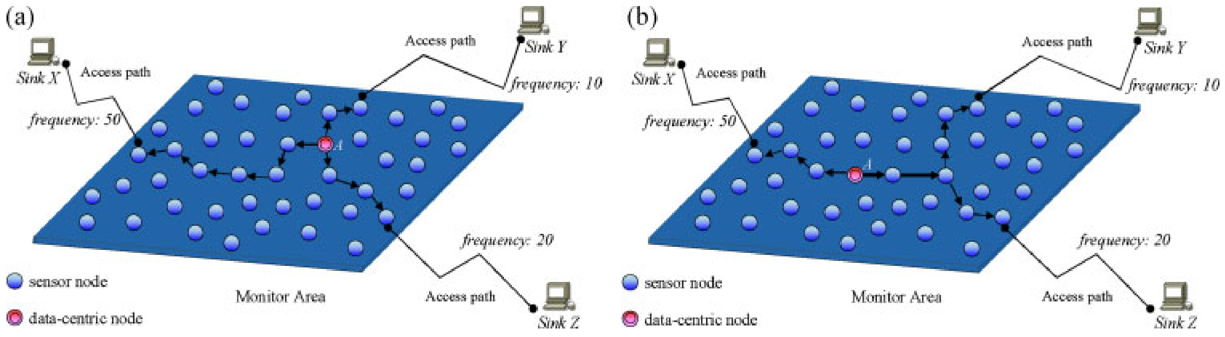


Fig. 1. An example to illustrate the obtained benefits if the location of the data centric node and the establishing a shared route are changed. (a) Static location of data-centric node *A* and routes by applying classic peer-to-peer algorithm. (b) Location of data-centric node *A* is dynamically changed, and a shared path is constructed from data-centric node *A* to multiple sink nodes.

mechanism dynamically changes the locations of the data-centric nodes, as shown in Figure 1(b). In Figure 1(b), the new location of the data-centric node *A* is closer to sink *X* due to the request of a high frequency. In comparison, Figures 1(a) and 1(b) respectively require 6 and 3 forwarding packets to reply to each data from the data-centric node to sink *X*. Although sinks *Y* and *Z* increase the forwarding overhead for replying data in Figure 1(b), the total cost of data reply decreases because their requests have low frequencies. A change in data-centric node will reduce the total forwarding overhead, especially for applications with long time data collection, conserving much energy for periodic data collecting.

Additionally, using the GPSR routing protocol to construct separate routes from data-centric nodes to each sink node results in a large amount of forwarding nodes that participate in the routes. Figure 1(a) depicts the individually constructed paths from the data-centric node *A* to each sink node. This paper proposes a path sharing routing protocol to construct a shared path from a data-centric node to multiple sink nodes. As shown in Figure 1(b), a shared path is constructed for sinks *Y* and *Z* according to their locations. Compared with Figure 1(b), Figure 1(a) depicts that several individual paths result in duplicate transmission, which consumes energy in forwarding nodes.

In summary, this paper proposes novel data-centric routing and storage mechanisms (DDCRS). The proposed routing mechanism automatically construct shared routes from data-centric nodes to multiple sink nodes in order for the efficient data collection. In addition, the proposed storage mechanism adaptively changes the locations of the data-centric nodes according to the locations and the reply frequencies of multiple sink nodes.

3. Dynamic Data-centric Routing and Storage Mechanisms

3.1. Protocol Architecture Overview

The proposed dynamic DCS mechanism mainly consists of two phases: the static phase and the dynamic phase, both of which are associated with different operations. Initially, a data-centric node is predefined using existing schemes [2,14–17]. In the static phase, the predefined data-centric node is responsible for storing the data transmitted from sensor nodes for replying with required data to the sink nodes. Herein, the data-centric node defined by a hash table is called a *home data-centric* (HD) node. Once a sensor node detects the event data, it transmits the data to the data-centric node that is closet to its location using the GPSR routing scheme [4]. The GPSR consists mainly of two algorithms for routing. One is a greedy forwarding algorithm which always forwards packets to the neighbor closest to the destination location. The other is a perimeter forwarding algorithm which uses right-hand rules to solve void areas when the greedy forwarding algorithm is impossible. Once the location of the data-centric node changes, the dynamic DCS mechanism will switch to a dynamic phase, handling both the data storage and the delivery problems.

As shown in Figure 2(a), the proposed mechanism operates first in the static phase, and then goes to the dynamic phase. In the static phase, when a sink node intends to send a query request to the data-centric node, it uses a pre-defined hash table to obtain the data-centric location of interested events and then sends the query packet using the GPSR routing scheme. The query packet contains information such as the frequency and duration of the data collection. Upon receiving the query packet, the HD node

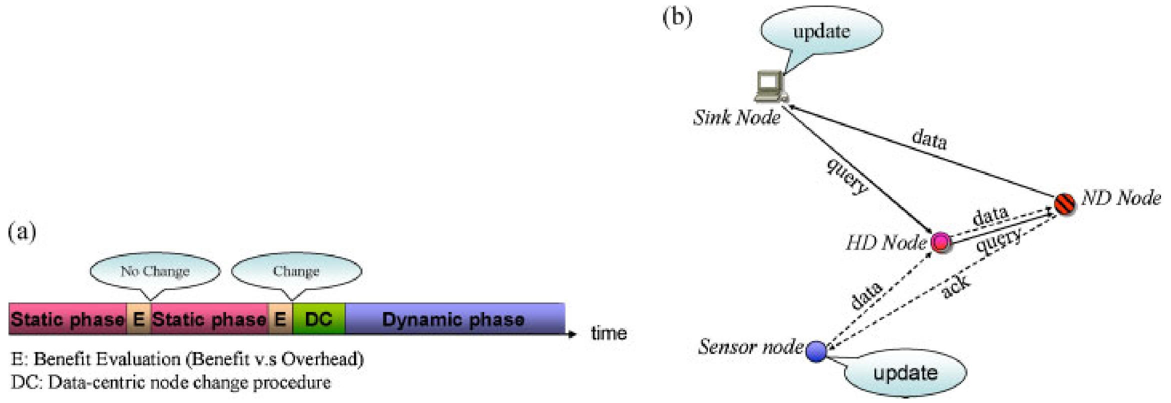


Fig. 2. Protocol architecture. (a) Static and dynamic phases of the proposed data centric storage mechanism. (b) Dynamic phase operations.

periodically replies with the request data using the proposed data-centric routing mechanism. Using to the locations of multiple sink nodes and their requested reply frequencies, the proposed data-centric routing mechanism constructs an efficient shared path for delivering the requested information with smaller communication overhead. If any sink node's query is overdue, the HD node stops to reply data to the sink node. Furthermore, the HD node executes a benefit evaluation to calculate the better location for the data-centric node and compares the benefit to the overhead of the changing data-centric node. In the case of it being worthwhile to change, the data-centric change procedure is executed, and the proposed dynamic DCS mechanism switches to the dynamic phase. Meanwhile, the HD node still acts as a data-centric node and the mechanism stays in the static phase.

When the data-centric node is changed, the proposed mechanism executes the operations defined in the dynamic phase. A simple but inefficient way to do this is to flood an update message to all sensor nodes and sink nodes, notifying them about the change in the data-centric node. However, blind flooding raises power consumption and thus the proposed mechanism has an alternative design to tackle this update notification problem. In the dynamic phase, the new data-centric node is called *new data-centric* node and is noted as ND for short. The HD node which is an old data-centric node is called *old data-centric* node and is noted as OD for short. In the meantime, the HD node is responsible for maintaining the location information of the ND node. Once the ND node changes again, the HD node maintains the up-to-date ND node's location information. Maintaining the ND node's location information makes the sink nodes' query and sensor nodes' store the correct information after data-centric node changing. Because the sink nodes and the sensor nodes are not aware of the change in data-centric nodes, they will send a query packet or sensed data to the HD node, as shown in Figure 2(b). Upon receiving the query or data packets, the HD node simply forwards the packets to the new data-centric node according to the ND node's location information it maintained. Upon receiving a query packet from the HD node, the ND node replies data to the sink node directly and notifies the sink node about the changes in the data-centric node. On the other hand, upon receiving the data packet from the HD node, the ND node stores the event data and sends an *Ack* to sensor nodes, indicating that the data have been successfully received and notifying the location of a new data-centric node. As sink nodes and sensor nodes are notified about the change in the data-centric node, they become aware of the dynamic phase. The sink and sensor nodes can then respectively query and store data to the ND node directly in the future.

The proposed mechanism notifies the sensor nodes and sink nodes about the location information of the ND node in a demanding manner; that is, the notification is made only when sensor nodes or sink nodes respectively intend to store or query the event data. This can efficiently reduce the control overhead and power consumption in order to maintain information on new data-centric node on sink and sensor nodes.

3.2. Data-centric Routing Mechanism

A routing protocol is required to establish a route to send data from a data-centric node to multiple sink nodes. This subsection describes a routing mechanism that constructs a shared path to reduce duplicated data transmission. All sink nodes that have sent requests might have different request frequencies. Therefore, the problem considered in this paper is similar to the generalized Steiner tree problem [18], which aims to minimize the sum of the weighted Euclidean distances. Since the generalized Steiner tree problem is an NP-hard problem [18], the computational complexities of the existing algorithms are too high to be executed in a sensor node which has limited computational ability. This paper proposes a heuristic data-centric routing mechanism which finds the forwarding nodes to construct the shared path in a distributed manner. The data-centric nodes and forwarding nodes can easily select the next forwarding nodes from their neighbors with low computational complexity.

For ease of description, some symbols are defined below. Let $d(A, B)$ denote the distance between nodes A and B. Let ShareGroup(s_1,s_2, $\cdots$,s_k) represent that k sinks (s_1,s_2, $\cdots$,s_k) can share the same path. Assume there are n sinks (s_1,s_2, $\cdots$,s_n) that request data from data-centric node D. Let f_{s_i} denote a query frequency of a sink s_i. Since each node is aware of its neighbors' location information, node D constructs a neighbor information table (NIT). Suppose that node D has m neighbors n_1,n_2, $\cdots$, n_m. As shown in Figure 3, in NIT, every entry n_i records the sink nodes in which n_i can efficiently forward packets. More specifically, if the distance $d(n_i, s_j)$ is smaller than the distance $d(D, s_j)$, data packets can be forwarded to sink s_j through neighbor n_i. Therefore, sink s_j will be recorded in the entry associated with n_i. Similar to node D, each node is able to construct its NIT. Let SinkNodeSet(n_i) be a function which returns a sink node set associated with the n_i in NIT. Let $w_i = \sum_{s_j \in \text{SinkNodeSet}(n_i)} f_{s_j}$.

Let p be the current forwarding node and p has m neighbors $n_1, \cdots, n_m$. Node p will select neighbor n_x to be the next forwarding node if $w_x > w_i$ for all $1 \leqq i \leqq m$. That is, the next forwarding node n_x should satisfy the condition $w_x = \max(w_1, w_2, \cdots, w_m)$. As a result, the constructed shared path from node p to the sink nodes has the maximal sum of frequencies. When the node D intends to reply data to the multi-sink, it takes the frequencies of interested sink nodes into account and constructs a shared path for sink nodes according to the Share_Path_Construction Algorithm described below. The algorithm selects a neighbor that can forward data to sink nodes with the maximum sum of frequencies until the selected neighbors can reply sensing information to all of the requested sink nodes. Since the route length of node D and the sink node with larger frequency is decreased by selecting the forwarding node that can send the data packet to the sink node with larger frequency, the total number of transmitted data packets can be reduced. When the forwarding node is selected, the data-centric node then broadcasts this information to its neighbors. Upon receiving the information, the forwarding nodes select their neighbors to play the role of forwarding nodes by similar operations done by the data-centric node. After that, the routing table can then be constructed in each forwarding node.

```
Algorithm: Share_Path_Construction (n, NIT)
Suppose node D's neighbors have their sink node sets, $k_1, k_2, \ldots, k_m$ in NIT, respectively.
Initial:
    ReplySink=∅
    SelectedNeighbor=∅
Begin
  while |ReplySink| < n
    select $n_x$ to be the next forwarding node,
        where $n_x$ satisfies $w_x = \max(w_1, w_2, \ldots, w_m)$ and $1 \leqq x \leqq m$
    insert SinkNodeSet($n_x$) into ReplySink set
    insert $n_x$ into SelectedNeighbor set
    remove SinkNodeSet($n_x$) from NIT
  end while
  Construct routing table with SelectedNeighbor set
End
```

Figure 3(a) depicts an example of constructing a shared path. Assume that $f_A > f_B = f_C > f_Z$. The data-centric node D has five neighbors, n_1, n_2, n_3, n_4, and n_5. Node D intends to reply data to sinks A, B, C, Z ($n = 4$) and the NIT information of node D is shown in Figure 3(a). Initially, set ReplySink $= \emptyset$ and SelectedNeighbor $= \emptyset$. Since the sink node set of neighbor n_3 has the maximal sum of frequencies, selecting n_3 as a forwarding node takes maximal advantage of the frequencies of the sink nodes A and B and the sharing path from D to the sink nodes A and B. Node D therefore selects n_3 as a forwarding node for sinks A and B. Node D then inserts n_3 in SelectedNeighbor set ($=\{n_3\}$) and adds sink nodes A and B into ReplySink set ($=\{A, B\}$). Since $|\text{ReplySink}| < n = 4$, some sink nodes require other forwarding nodes to forward the data. Node D then removes $\{A, B\}$ from NIT and repeats the above steps. Subsequently, neighboring nodes n_2 and n_1 are selected to forward data from D to sinks C and Z, respectively. As a result, ReplySink $= \{A, B, C, Z\}$ and $|\text{ReplySink}| = 4$, indicating that node D has found forwarding nodes for all sink nodes and finishes the algorithm. After that, node D records the Selected-Neighbor $= \{n_3, n_2, n_1\}$ as a routing table and sends a path construction request to the three nodes. Since sinks A and B share the path from D to the common forwarding node n_3, we denote the relation of path share information by ShareGroup(A, B), (C), (Z). Upon receiving the path construction request, nodes n_1, n_2, n_3 execute the same algorithm to further construct a share path for the sink nodes in ShareGroup. Finally, as depicted in Figure 3(a), node T individually sends each packet to sinks A and B because no neighbor of T can forward packets to both sinks A and B efficiently. While the whole shared path is constructed, the data-centric node D periodically sends the requested data to multiple sinks according to the shared path, which is recorded in the routing table of each forwarding node.

Some other complicated case may occur since it is possible that two neighbors can forward to the same sink at the same time. As shown in Figure 3(b), data-centric node D has six neighbors, n_1, n_2, n_3, n_4, n_5, and n_6. The above mentioned algorithm will select n_3, n_4, and n_1 as the forwarding nodes where either nodes n_3 or n_4 could be the forwarding node from D to sink A. Therefore, two results of shared routes are possible: (1) ShareGroup(A, C), (B), (Z); (2) ShareGroup(A, B), (C), (Z). To avoid duplicate transmissions of the same data packet to the same sink node, the cost of two shared routes are compared, wherein the smaller one is selected. Regarding the cost calculation of a shared route, the concept of the shared degree is introduced below. The degree of path sharing of two sink nodes, say s_i and s_j, is defined by the common path length of the two sinks and is denoted by Sd(s_i, s_j). Let symbol α_{ij} denote the angle $\angle s_i D s_j$. In fact, the angle α_{ij} determines the shared degree of sink nodes s_i and s_j. The larger the angle of α_{ij} is the smaller the shared degree of nodes s_i and s_j becomes. Therefore, the value of a shared degree could be estimated using the following formula which normalizes the value between 0 and 1.

$$\mathrm{Sd}(s_i, s_j) = \begin{cases} 1 & , \text{ if } \quad \alpha_{ij} = 0^\circ \\ 1 - \frac{\alpha_{ij}}{180^\circ} & , \text{ if } \quad 0 < \alpha_{ij} < 180^\circ \\ 0 & , \text{ if } \quad \alpha_{ij} = 180^\circ \end{cases} \quad (1)$$

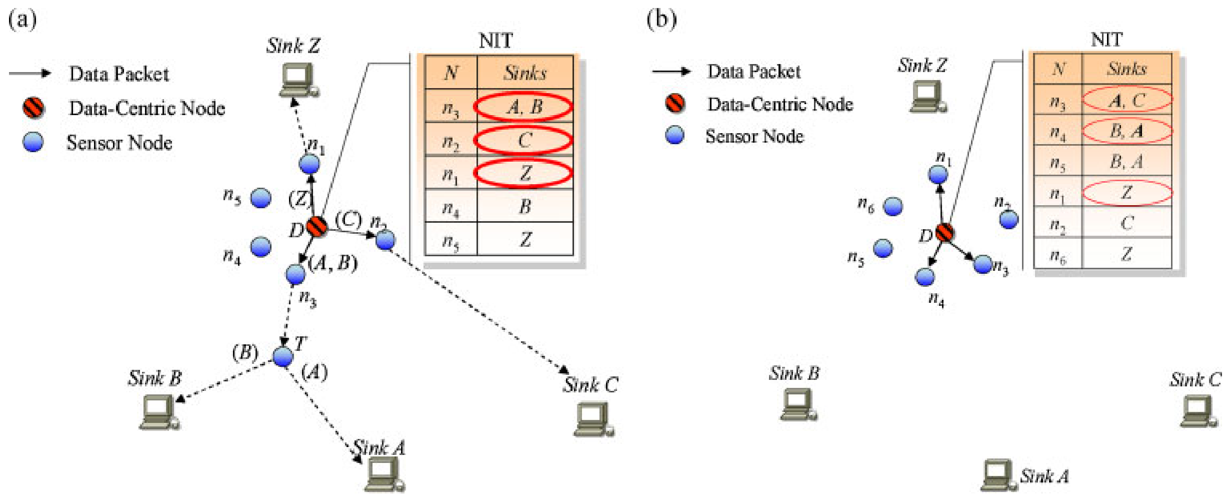


Fig. 3. Data-centric routing mechanism. (a) An example of constructing a shared path from node D to sinks A, B, C, and Z. (b) An example that two neighbors n_3 and n_4 are chosen to forward data to the same sink node A.

The shared degree can be used to estimate the cost of the shared routing path. Besides, in order to accurately estimate the cost of the shared route, the frequencies of two sink nodes are considered in the calculation of the median point of the two sink nodes. Assume that $f_{s_i} > f_{s_j}$. As shown in Figure 4(a), a median point E with coordination $\left(x_1 f_{s_i} + x_2 f_{s_j} / f_{s_i} + f_{s_j}, y_1 f_{s_i} + y_2 f_{s_j} / f_{s_i} + f_{s_j}\right)$ of s_i and s_j can be found. The end point F of the shared path can then be calculated by following vector evaluation using the location information of $\mathrm{Sd}(s_i, s_j)$ and point E below. First, $d(D, E)$ and vector $\overrightarrow{DE}$ are calculated using Equations (2) and (3)

$$\left|\overrightarrow{DE}\right| = \sqrt{\left(\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{si} + f_{s_j}} - x\right)^2 + \left(\frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y\right)^2} \tag{2}$$

$$\overrightarrow{DE} = \left(\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{s_i} + f_{s_j}} - x, \frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y\right) \tag{3}$$

Then, the unit vector $\vec{u}$ can be obtained by the following equation:

$$\vec{u} = \frac{\overrightarrow{DE}}{\left|\overrightarrow{DE}\right|} = \left(\frac{\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{s_i} + f_{s_j}} - x}{\sqrt{\left(\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{s_i} + f_{s_j}} - x\right)^2 + \left(\frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y\right)^2}}, \frac{\frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y}{\sqrt{\left(\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{s_i} + f_{s_j}} - x\right)^2 + \left(\frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y\right)^2}}\right) \tag{4}$$

Hereafter, vector $\overrightarrow{DF}$ can be calculated by shared degree $\mathrm{Sd}(s_i, s_j)$ and unit vector $\vec{u}$, as shown in Equation (5):

$$\overrightarrow{DF} = \vec{u} \cdot \mathrm{Sd}(s_i, s_j) \cdot \left|\overrightarrow{DE}\right| = \left(\frac{\left(\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{s_i} + f_{s_j}} - x\right) \cdot \mathrm{Sd}(s_i, s_j) \cdot \left|\overrightarrow{DE}\right|}{\left|\overrightarrow{DE}\right|}, \frac{\left(\frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y\right) \cdot \mathrm{Sd}(s_i, s_j) \cdot \left|\overrightarrow{DE}\right|}{\left|\overrightarrow{DE}\right|}\right) \tag{5}$$

Finally, F can be calculated with Equation (6).

$$F = \left(\frac{\left(\frac{x_1 f_{s_i} + x_2 f_{s_j}}{f_{s_i} + f_{s_j}} - x\right) \cdot Sd(s_i, s_j) \cdot \left|\overrightarrow{DE}\right|}{\left|\overrightarrow{DE}\right|} + x, \frac{\left(\frac{y_1 f_{s_i} + y_2 f_{s_j}}{f_{s_i} + f_{s_j}} - y\right) \cdot Sd(s_i, s_j) \cdot \left|\overrightarrow{DE}\right|}{\left|\overrightarrow{DE}\right|} + y\right) \tag{6}$$

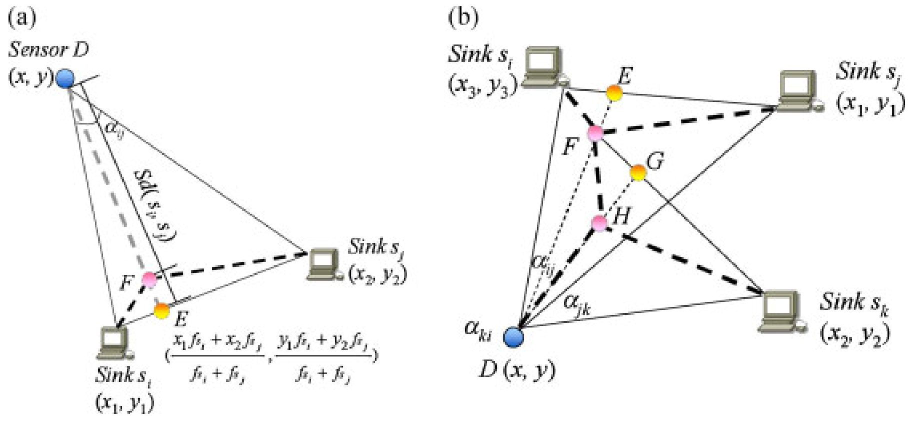


Fig. 4. An example illustrating how to evaluate the cost of a shared path. (a) An example that two sinks share a common routing path. (b) An example that a route shared by three sinks; sinks s_i, s_j, and s_k share segment $\overline{DH}$ and then sinks s_i and s_j additionally share segment $\overline{HF}$.

The point F is a branch point of shared path of s_i and s_j. Given a query frequency f_{s_i} and f_{s_j} of sinks s_i and s_j, respectively, the cost of ShareGroup(s_i, s_j) is estimated by RouteCost, as shown in Equation (7), which calculates the number of packets generated on the path for delivering data packets to sinks s_i and s_j. In Equation (7), the $d(D, F)$ is the shared segment length of s_i and s_j and the cost $\text{Max}(f_{s_i}, f_{s_j})$ is taken into account. The $d(s_i, F)$ and $d(s_j, F)$ are the segment lengths that are not shared by s_i and s_j. The costs of $d(s_i, F)$ and $d(s_j, F)$ are f_{s_i} and f_{s_j}, respectively. The cost of a shared path of sinks s_i and s_j can therefore be measured by the following equations:

$$\text{RouteCost} = \text{Max}(fs_i, fs_j) \times d(D, F) + fs_i \times d(s_i, F) + f_{s_j} \times d(s_j, F) \tag{7}$$

In case there are more than two sinks, say $s_1, s_2, \cdots, s_k$, the same data packet can be shared, and the route cost can be calculated in the order of the frequencies of sinks, from large to small. The shared route cost of two sink nodes with the first two high frequencies will be calculated first and then their shared point and the sink with higher frequency will be executed the same operations until all sink nodes are calculated. As Figure 4(b) depicts, sinks s_i, s_j, and s_k share the same data packet. Suppose that $f_{s_i} > f_{s_j} > f_{s_k}$. The shared route cost of sinks s_i and s_j is first calculated. Then the shared route cost of the share point F of s_i and s_j and s_k is calculated. The final shared path has sinks s_i, s_j, and s_k sharing segment $\overline{DH}$ and sinks s_i and s_j additionally sharing segment $\overline{HF}$, as shown in Figure 4(b). The location of median point E can be calculated by the locations and frequencies of sinks s_i and s_j and then the location of point F can be derived by applying Equation (6). With this, the locations of points G and H can also be obtained. Consequently, the routing cost of ShareGroup(s_i, s_j, s_k) is

$$\text{RouteCost} = \text{Max}(fs_i, fs_j, fs_k) \times d(D, H) + fs_k \times d(H, S_k) + \text{Max}(fs_i, fs_j) \times d(H, F) + fs_i \times d(F, s_i) + fs_j \times d(F, S_j) \tag{8}$$

In Equation (8), $d(D, H)$ is the shared length of s_i, s_j, and s_k and $d(H, F)$ is the shared length of s_i and s_j. There are costs $\text{Max}(fs_i, fs_j, fs_k)$ and $\text{Max}(fs_i, fs_j)$ on the two shared segments, respectively. The $d(H, s_k)$, $d(F, s_i)$, and $d(F, s_j)$ are the length of non-sharing paths, with their frequencies being fs_k, fs_i, and fs_j, respectively.

3.3. Dynamic Data-centric Storage Mechanism

This subsection proposes a dynamic DCS mechanism that dynamically determines a better location of a data-centric node according to sink nodes' location and requested data collection frequency. The dynamic DCS mechanism consists of a static phase and a dynamic phase, which are used according to whether or not the HD node plays the role of the data-centric node. Initially, a predefined HD node determined by the hash mechanism plays the role of the data-centric node, responsible for storing event information sent from sensor nodes. When the data-centric node receives a new query packet or when the old query is overdue,

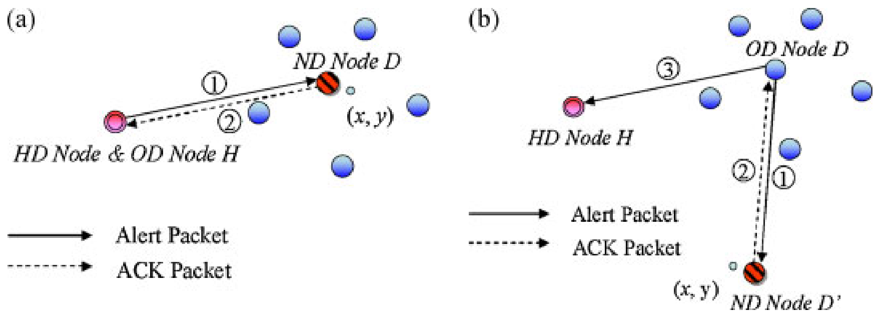


Fig. 5. Data-centric node change procedure. (a) Static phase. (b) Dynamic phase.

it executes a benefit evaluation to estimate the benefit and overhead obtained from changing the location of the data-centric node. Before estimating the benefit and overhead, the better location of a data-centric (ND) node is derived. Suppose there exist n sink nodes s_i located at (x_i, y_i), where $i = 1, 2, \cdots, n$ and they query the same data-centric node for data collection. The new data-centric node should be closer to the sink that has a higher frequency of request. This will reduce the cost for replying data to the sink nodes. Therefore, the median point evaluated based on the locations and frequencies of all sink nodes will be the better location of a data-centric node. Equation (9) reflects this concept. An OD node can derive the better location (x, y) of a data-centric node by using the following equation:

$$\begin{cases} x = \dfrac{\sum_{i=1}^{n} x_i f_i}{\sum_{i=1}^{n} f_i} \\ y = \dfrac{\sum_{i=1}^{n} y_i f_i}{\sum_{i=1}^{n} f_i} \end{cases}, \quad \text{where } f_i \text{ is sink } i\text{'s report frequency} \tag{9}$$

The OD node then estimates the benefit and the overhead of changing the data-centric node. In case it is worthwhile to change the data-centric node, the OD node sends an *alert* packet to find *ND* an node closest to the location (x, y) using the GPSR [4] routing protocol. Upon receiving an *alert* packet, the ND node replies to the OD node with an *Ack*. The OD node then starts to transmit all event data and sink query information to the ND node. After the data-centric change procedure is finished, the ND node takes the place of the OD node and then now becomes responsible for the reply data to the sink nodes.

Figure 5(a) is an example that illustrates the operation of changing the data-centric node in the static phase. Initially, a pre-defined data-centric node H (HD node) is responsible for storing the data of an event type. The HD node firstly calculates the location (x, y) of the new data-centric node, and then evaluates the benefit and overhead of changing a data-centric node from H to its new location (x, y). In case the HD node determines that it is worthwhile to change the location of the data-centric node, node H transmits an *alert* packet to find node D which is closest to location (x, y), as shown in Step 1 of Figure 5(a). As node D receives the alert packet, it replies with an *Ack* packet, as shown in Step 2 of Figure 5(a). Node H then transmits all event data and sink query information to the ND node D. By then, the ND node D is now responsible for replying data to the sink nodes, while the HD node H maintains the ND node's location information. The change of data-centric nodes initiates the data-centric mechanism, thereby entering the dynamic phase.

Figure 5(b) is an example where the data-centric node changes in the dynamic phase. Assume that the data-centric node has changed from a HD node H to node D. Once the benefit evaluation determines to change the location of the data-centric node from node D to location (x, y), node D will then find the ND node, say D', and send an *alert* packet to D', as depicted in Steps 1 and 2 of Figure 5(b). After that, node D sends an update packet to HD node H according to the hash table to register the location information of ND node D', as depicted in Step 3. The benefit evaluation will be explained in detail in the next subsection.

In the dynamic phase, the data-centric node changes from a HD node to a ND node. However, the new sink that never queried event information from a data-centric node will be not aware of the change. In addition, a data-centric node may be changed several times between two successive queries. This also makes sink nodes unable to maintain the locations of new data-centric nodes. Problems raised from the change of data-centric nodes can be categorized into the following two cases.

3.3.1. New sink query

In the dynamic phase, sink nodes that have never queried the event before are not aware that the

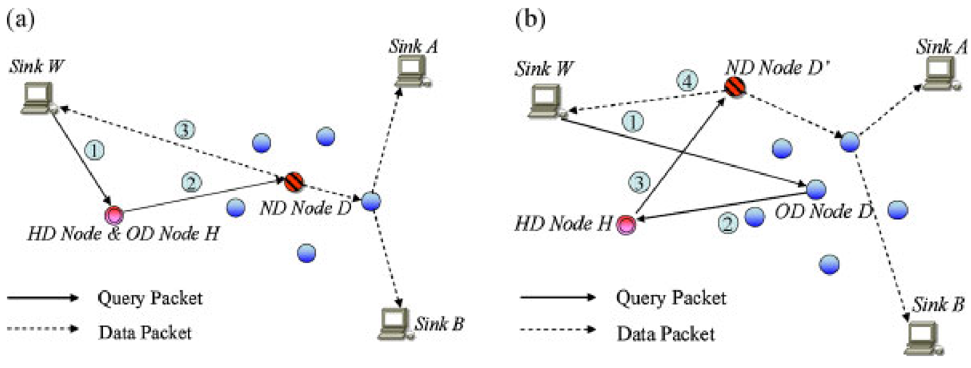


Fig. 6. Examples that illustrate the query forwarding update procedure. (a) Operations when the first new sink queries. (b) Operations when data-centric node has been changed several times during two consecutive queries.

data-centric node has been changed. Therefore, these sink nodes will use the hash table and will send their queries to the HD node. When the HD node receives the sinks' queries, it forwards the query packets to the ND node because its location information is maintained. Upon receiving the query packet, the ND node executes data-centric routing mechanism to construct a shared path from the ND node to all interested sink nodes, including the new sink nodes, and then replies the data to sink nodes according to their requests and updates its location information to these sink nodes for the event type. When sink nodes receive the location update information from the ND node, they become aware that the data-centric node has been changed to the ND node, and will send their query requests next time to the ND node instead of the HD node.

Take Figure 6(a) as an example. Suppose the data-centric node has been changed from the HD node H to the ND node D, and the node D is replying data to sinks A and B. At this moment, a new sink W intends to query the data-centric node D. Because sink W executes the query the first time, it uses a hash table to obtain the location of the HD node, and then the sends query to HD, as shown in Step 1 of Figure 6(a). Since the HD node maintained the ND node's information, it knows that the data-centric node has been changed to node D. The HD node then forwards the query to the ND node D as shown in Step 2 of Figure 6(a). Upon receiving the query, node D then uses the data-centric routing mechanism to construct a shared path for replying data periodically to sinks A, B, and W and sends W an update message which contains the location of the ND node D so that sink W can query node D directly next time, as shown in Step 3 of Figure 6(a).

3.3.2. *Data-centric node changes several times*

In the dynamic phase, the data-centric node could be changed several times between two successive queries of a sink node, causing the sink node to maintain a wrong location of the data-centric node. Assume a sequence of nodes $d_0 =$ HD node, $d_1, d_2, \ldots, d_x =$ ND node have played the role of the data-centric node successively. Assume that the location of the data-centric node maintained by a sink node, say W, is d_i and the sink W intends to query information. Therefore, sink W sends a query to d_i. In case $i < x$, node d_i is an OD node of the event and the location of the data-centric node maintained by sink W is wrong due to the frequent change of data-centric node during two successive queries of sink W. Upon receiving the request packet, the OD node d_i forwards the query to the HD node d_0, and node d_0 forwards the query to correct the ND node d_x directly. The reason for this design is that OD node d_i cannot guarantee that its next node in sequence is the correct ND node even though it has maintained the location information of the next ND node d_{i+1}. Since the new ND node always notifies its location information to the HD node, the HD node maintains an up-to-date location information of ND node d_x. Therefore, as the HD node d_0 receives the query forwarded by the OD node d_i, it forwards the query to the ND node according to the information it maintained.

Take Figure 6(b) as an example. Assume sink W has sent a query to the data-centric node D previously. After that, the data-centric node changes from node D to node D'. As sink W attempts to query the information, it sends a query packet to node D according to the maintained location of the data-centric node, as shown in

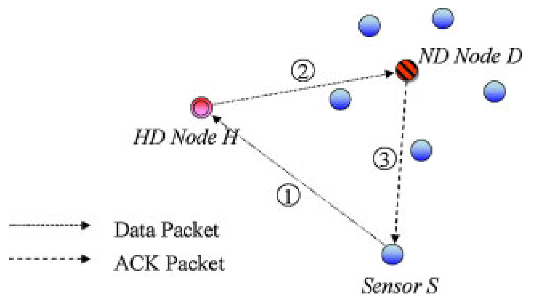


Fig. 7. Data forwarding update procedure.

Step 1 of Figure 6(b). Upon receiving the query from sink *W*, node *D* forwards the query to the HD node in order to obtain the location of the ND node *D′*, as shown in Step 2 of Figure 6(b). The HD node then forwards the query to the ND node *D′*, as shown in Step 3 of Figure 6(b). Upon receiving the query, the ND node *D′* replies to sink *W* the sensing information according to the requested data collection frequency. The sink node *W* then updates its location information of the data-centric node.

In addition to sink nodes, the change of the data-centric node also makes sensor nodes maintain the wrong location information. Operations designed in the proposed mechanism for sensor nodes are similar to sink nodes as described previously. When a sensor node intends to store event data to the data-centric node, it transmits the data packet to the HD node, as shown in Step 1 of Figure 7. After that, the HD node forwards the data packet to the ND node for data storing, as shown in Step 2 of Figure 7. Additionally, the ND node sends an *Ack* packet to the sensor node to update its location information of new data-centric node, as shown in Step 3 of Figure 7.

3.4. Benefit Evaluation

There are two cases when the data-centric node will be initiated to calculate the location of the new data-centric node. One case is when the data-centric node receives a new query and the other case is when the old query is expired. To determine whether or not it is worthwhile to change the location of the data-centric node, the benefit and the overhead of changing the data-centric node from the OD node to the ND node are estimated and compared. Moreover, frequent changing of the data-centric node will result in a high overhead. Hence, in the estimation of data-centric node change, benefit evaluation considers the following three conditions:

(1) Is the remaining time of the old query long enough?
(2) Is the new query's duration of data collection long enough?
(3) Is the benefit obtained from the change of data-centric node larger than the overhead?

Consider the conditions of (1) and (2). Suppose that the OD node has replied data to *n*-1 sinks, s_1, s_2, …, s_{n-1}. Assume that the OD node receives a new query from sink s_n. In addition, assume that the remaining duration of queries of $s_1, s_2, \ldots, s_n$ are t_1, $t_2, \ldots, t_n$, respectively, and t_1 $t_2, \ldots, t_{n-1}$. In case $t_n > t_{n-1} > t_{\text{threshold}}$, it shows that the shortest remaining duration is long enough for the change of the data-centric node, where $t_{\text{threshold}}$ is a threshold value of remaining duration of the query. On the other hand, if $t_{n-1} < t_{\text{threshold}}$, it means that the remaining duration of the existing query is too short. Therefore, it is unnecessary to change the data-centric node for this new query. Consider the second condition. In case $t_{n-1} > t_n > t_{\text{threshold}}$, it shows that the duration of a new query is long enough to change the data-centric node. On the contrary, the duration of the new query is too short to change the data-centric node. Consequently, we can determine whether it is worthwhile to change data-centric node for time constraint using the following rule:

Time constraint rule:

```
Time constraint rule:
   Let $t_{\min}$=min ($t_1, t_2, \ldots, t_n$).
   If ($t_{\min} > t_{\text{threshold}}$)                /* worthwhile to change */
      Call Benefit_Overhead_Evaluation() /*described later */
Else
      No change for data-centric node due to the duration is too short.
```

Even though conditions (1) and (2) are satisfied, condition (3) should be verified to guarantee that the benefit is larger than the overhead obtained from the change of the data-centric node. Firstly, let Cost(data-centric node) denote the routing cost of the data-centric node which needs to reply data to sink nodes. From the statement in Section III C, it costs less to reply data to sink nodes if the data-centric node changes to the median point of the querying sink nodes. Changing the data-centric node to the median point of the querying sink nodes can get the benefit Bnt = Cost(OD node) − Cost(ND node), where the calculations of Cost(OD node) and Cost(ND node) could be obtained by Equation (7). However, an angle threshold α is used herein to predict the benefit obtained from the shared paths between two sink nodes. If the angle between sinks s_i and s_j is smaller than angle threshold α, the cost of sinks s_i and s_j is calculated by ShareGroup(s_i, s_j). Otherwise, the costs of s_i and s_j are calculated by their individual path.

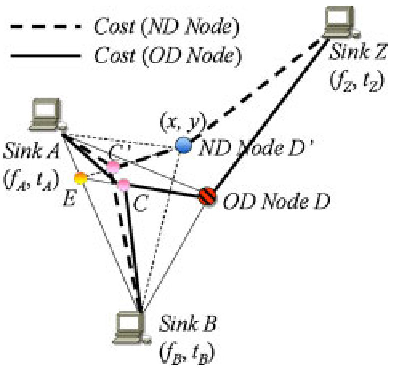


Fig. 8. An example for illustrating benefit evaluation.

Take Figure 8 as an example to illustrate benefit evaluation. Suppose that sinks A, B, Z request for data collection with frequencies f_A, f_B, and f_Z, respectively, and the OD node D calculates the median points (x, y) of sinks A, B, Z as depicted in Figure 8. In case the angle $\angle$ADB is smaller than the predefined threshold α and $f_A > f_Z > f_B$, sinks A and B are expected to share the same path $d(D, C)$. Let E be the median point of A and B. The location of point C can be obtained by applying Equation (6). Cost(D) can then be evaluated by applying Equation (7). Similarly, Cost(D') can also be obtained. Therefore, the benefit Bnt of changing the data-centric node from the OD node to the ND node is estimated by Cost(D)-Cost(D'). Discussion of how to set the angle threshold will be investigated in simulation.

The overhead of the data-centric node change from the OD node to the ND node could be evaluated by the cost when the OD node transmits event data and sinks' information to the ND node. The overhead, denoted by O, can therefore be evaluated by Expression (10), where data are the total data size of the event data stored in the OD node and sinks' information.

$$O = \text{Data} \times d(\text{OD}, \text{ND}) \tag{10}$$

After calculating the benefit Bnt and the overhead O, the following policies can be used to determine whether or not it is worthwhile to change the data-centric node's location.

$$T_{\min} \times \text{Bnt} - O > \text{Threshold} \tag{11}$$

where $T_{\min}$ denotes the minimal query remaining time of all sink nodes mentioned in conditions (1) and (2). If Criterion (11) is satisfied, the data-centric node change procedure described previously is executed. The developed mechanism then switches to the DC stage, as depicted in Figure 2(a).

4. The Protocol and Related Issues

This section presents the complete protocols. In addition, other issues about failure and the power balance of home nodes and data-centric nodes and the latency of the protocol are discussed.

4.1. The Protocol

The proposed DDCRS operates in an event-driven manner. Three procedures, namely the *insertion procedure*, the *lookup procedure*, and the *data-centric node change procedure* are designed to handle their the corresponding events. The insertion procedure is executed when the sensor nodes intend to store the detected event data. The lookup operation procedure is executed as sink nodes attempt to query event types from the data-centric node. The data-centric node change procedure is then executed as the location of the data-centric node of an event type changes. The algorithms of the three procedures are described as follows.

4.1.1. Insertion procedure

As depicted in Figure 9(a), when a sensor node detects event types, it initially obtains the location of the HD node from hash table and sends a data packet to the HD node for storing by using the GPSR [4] routing protocol. In case the mechanism stays in the static phase, the HD node becomes the data-centric node of the event. Upon receiving the event data from sensor node, the HD node stores the event data in its local memory. If the mechanism stays in the dynamic phase, the HD node forwards the data packet to correct location of the new data-centric node, i.e. the ND node, according to the maintained location information. When the ND node receives the data packet, it stores the event data and sends an *Ack* packet to the sensor node to update the new location of the data-centric node. Subsequently, the sensor node can directly send detected event data to the ND node next time.

4.1.2. Lookup procedure

As depicted in Figure 9(b), when a sink node attempts to query an event, the lookup procedure is initiated. Initially, the sink also obtains the location information of the HD node using the hash table. It sends a query packet that includes the event type, the data collection frequency, and the query duration. In case the HD is the data-centric node of the event type (static phase), the HD node executes a benefit evaluation to determine

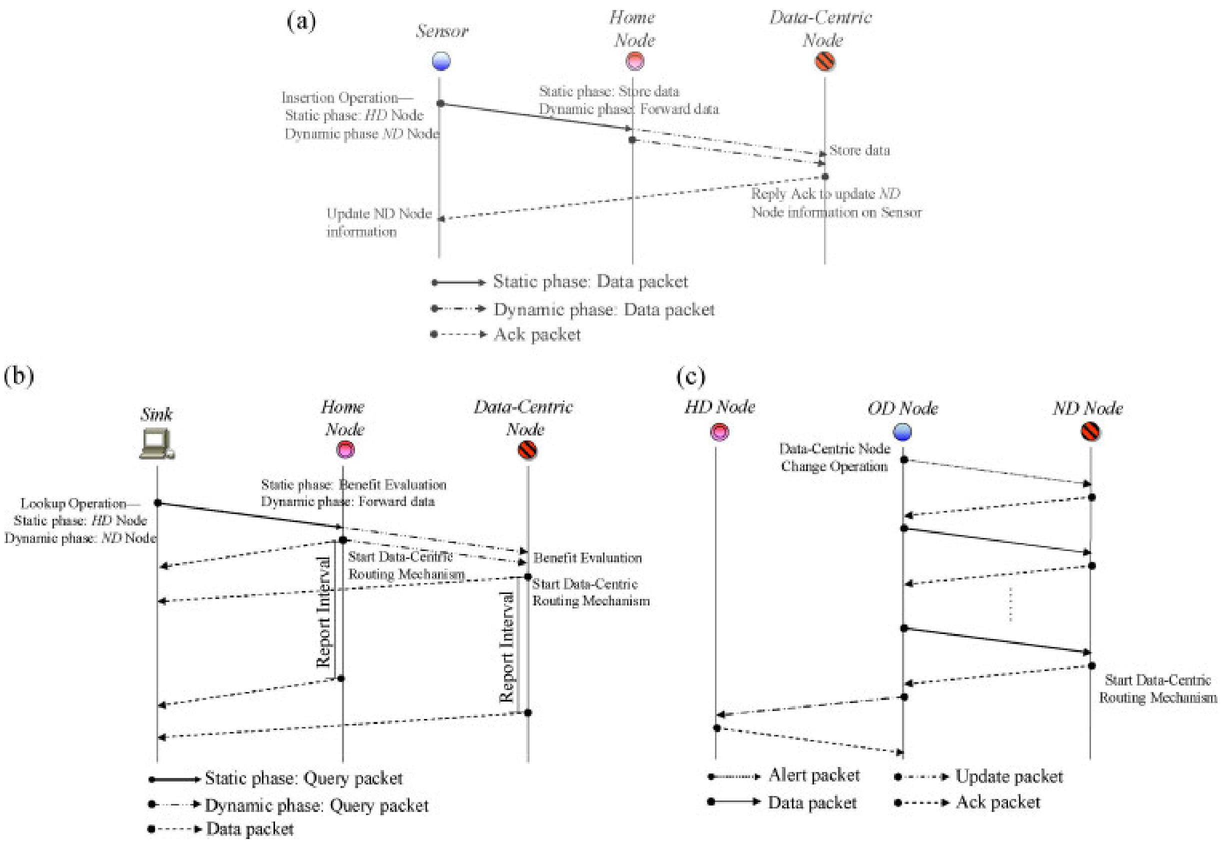

(a)
Sensor
Home Node
Data-Centric Node
Insertion Operation— Static phase: HD Node Dynamic phase ND Node
Static phase: Store data
Dynamic phase: Forward data
Store data
Reply Ack to update ND Node information on Sensor
Update ND Node information
Static phase: Data packet
Dynamic phase: Data packet
Ack packet
(b)
Sink
Home Node
Data-Centric Node
Lookup Operation— Static phase: HD Node Dynamic phase: ND Node
Static phase: Benefit Evaluation
Dynamic phase: Forward data
Start Data-Centric Routing Mechanism
Benefit Evaluation
Start Data-Centric Routing Mechanism
Report Interval
Report Interval
Static phase: Query packet
Dynamic phase: Query packet
Data packet
(c)
HD Node
OD Node
ND Node
Data-Centric Node Change Operation
Start Data-Centric Routing Mechanism
Alert packet
Update packet
Data packet
Ack packet

Fig. 9. Protocol operations. (a) Insertion operation procedure. (b) Lookup operation procedure. (c) Data-centric node change procedure.

whether or not the *data-centric node change procedure* should be initiated. If the data-centric node changes, the new data-centric node becomes responsible with the reply data; otherwise, the data-centric routing mechanism is applied to construct a shared path for replying data to the sink nodes according to their requests. On the contrary, if the HD node is not the data-centric node (dynamic phase), the HD node then forwards the query to the ND node according to the maintained location of the ND node. Upon receiving the query packet, the ND node executes data-centric routing mechanism. Similarly, the ND node sends an *Ack* packet to the sink node, where it tries to update the new location of the ND node.

4.1.3. Data-centric node change procedure

As shown in Figure 9(c), when the data-centric node receives a new query packet from some sink node, or when an old query is expired, benefit evaluation, described in Section 3, is executed. In case Criterion (11) is satisfied, the data-centric node finds the sensor node closest to the median point and sends an *alert* packet to the sensor using the GPSR [4] routing protocol. Upon receiving the *alert* packet from a data-centric node, the new data-centric node replies with an *Ack* packet to the old data-centric node. The data-centric node then becomes an OD node of the event type and then starts to send event data and sink node query information to the ND node. As the data-centric node change procedure finishes, the OD node sends an *update* packet to the HD node to maintain location information of the ND node. Consequently, the new data-centric node takes the place of the old data-centric node to execute the data-centric routing mechanism for replying data to the querying sink nodes.

The pseudo code for DDCRS is listed in Figure 10.

4.2. Failure and Power Balance Issues

A previous work [2] proposes that a failure handling mechanism can be similarly applied to the proposed DDCRS mechanism as the failure of home node and data-centric nodes occurs. Since GPSR [4] can

```
Sensor Node—
Insertion_procedure (data)
 1 if Static_phase
 2   then send data to DHT(data) location
 3 else
 4   send data to updated location
 5 end if

Sink Node—
Lookup_procedure (query)
 1 if Static_phase
 2   then send query to DHT(query) location
 3 else
 4   send query to updated location
 5 end if

Data-Centric Node—
Dynamic_Data-Centric_protocol( )
 1 switch (operation)
 2   case receive_query :
 3     if Static_phase
 4       then
 5         Data_Centric_Routing_Protocol( )
 6         if Benefit_Evaluation( ) is worthy to change
 7           then Change_Data-Centric_procedure( )
 8         end if
 9     else
10         Forward_Packet_procedure(query)
11     end if
12   break
13   case receive_data :
14     if Static_phase
15       then store data
16     else
17       Forward_Packet_procedure(data)
18     end if
19   break
20 end switch
```

Fig. 10. Dynamic data-centric routing and storage mechanism pseudo code.

find backup nodes near the data-centric node, once a data-centric node fails, the backup node closest to the location of the data-centric node becomes the new data-centric node, storing the new event data and serving the multi-sink nodes. Moreover, if the ND node fails in executing the *data-centric node change procedure*, the OD node can send a *repair* packet to find the backup node closest to the location of the ND node. Upon receiving the *repair* packet, the backup node replies with an *Ack* packet to the OD node and becomes the ND node of the event type. The *Ack* includes the information of the event data it backed up. Hence, the OD node can continue to send the event data that ND node does not store without retransmitting all of the event data.

In addition, if a home node or a data-centric node discovers that it is not appropriate to be the home node or data-centric node of an event type due to low power energy, it can broadcast a *failure* packet to inform all of its neighbors so that they can remove the data-centric node's information from the NIT. The backup nodes then start to select a new home or data-centric node whose location is closest to the data-centric location of the event type. The new home or data-centric node would then be able to operate correctly.

4.3. Latency Issue

The proposed dynamic DCS mechanism will increase latency for reporting data to the sink nodes when the data-centric node is changed to another node. There are two reasons for increasing the latency. One reason is that the old data-centric node transmitting the event data and the query information of sink nodes to the new data-centric node increases the latency. The other reason is that the new query of the sink node might be sent to the old data-centric node. When this happens, the old data-centric node forwards the new query to the home node which records the location of the new data-centric node. The home node then sends the new query to the new data-centric node. Upon receiving the data packet from the new data-centric node, the querying sink node records the location of the new data-centric node so that next query of the sink node can be directly sent to the new data-centric node.

5. Performance Study

This section investigates the performance of the proposed DDCRS. The following first describes the simulation environment then shows the investigated simulation results.

5.1. Simulation Model

The proposed DDCRS mechanism was implemented in GloMoSim (version 2.03) [19] and is compared with four storage mechanisms: LS, ES, DCS [2] and double ruling [10] (DR in short). In the LS mechanism, each source stores the event information in its local memory. Whenever the source receives queries from multiple sink nodes, it constructs a shared path to the sink nodes. In the ES mechanism, the source directly transmit the detected event information to all sink nodes, even though the sink nodes are not interested in the events. The radio range of each node is set to 80 m. Sensor nodes are uniformly and randomly distributed and the node density is controlled by a

Table I. Simulation parameters.

Parameters	Value
Node density (1/m^2)	1/1024
Radio range (m)	80
Total number of event types	9
GPSR beacon interval (s)	1
GPSR beacon expiration (s)	5
Planarization	GG
Simulation time (s)	420
Number of detected data in each event type	300
Number of sink nodes	3
Query generation rate (qps)	1/10
Shared path angle threshold (°)	60
Time constraint limit (s)	1

constant value of 1/1024, in which the average number of neighbors is 8. There are three sink nodes in the WSN at the corner of the monitoring square area. The requested data collection frequencies of the three sink nodes are set to 1/10, 1/20, and 1/40 dps (data per second), respectively. The query generation rate of each sink node is 1/10 qps (query per second).That is, each sink node sends a query every 10 s. In the WSN, nine event types and 300 data in each event type possibly are detected by the sensor nodes. Each sensor node has the same probability of event detection. Related parameters of the simulation are listed in Table I.

The simulation uses three metrics to evaluate the performance of the mechanisms: (1) total messages—the total number of packets in the WSN; (2) hotspot usage—the maximum number of packets to cross any signal link; (3) fairness index—Jain's fairness index [20] of traffic for all sensor nodes. The Jain's fairness index normalized between 0 and 1 is defined by the following equation:

$$\text{Jain's fairness index} = \frac{\left[\sum_{r=1}^{t} p_r\right]^2}{t\left[\sum_{r=1}^{t} (p_r)^2\right]} \qquad (12)$$

where t denotes the total number of sensor nodes and p_r denotes the traffic for sensor node r. In case all p_r have same value, the result of the fairness index is equal to 1, which is an optimal value. The fairness index of a mechanism approaching 1 indicates that the mechanism provides a better fairness.

Each sink node randomly selects an event type as its query interest, but the number of event types that each sink can query is under control. In addition to implementing LS, ES, DCS [1], and DR[10] mechanisms, the DDCS and DDCRS mechanisms developed in this paper are implemented. The DDCS adopts only the dynamic DCS mechanism without involving the data-centric routing mechanism. That is, the DDCS constructs individual paths from the data-centric node to each sink. The DDCRS adopts both dynamic DCS and data-centric routing mechanisms to construct shared paths when a data-centric node replies data to multiple sinks. In the DDCS and DDCRS mechanisms, a data-centric node considers executing the dynamic DCS mechanism only when it receives queries from more than one sink node. In other words, if the data-centric node only receives one query from a sink node, it keeps operating in the traditional DCS [1] mechanism. The DCS, DR, DDCS, and DDCRS mechanisms belong to DCS-based mechanisms. Each result is obtained from an average of 10 experiments. The 95% confidence interval is always smaller than ± 5% of the reported values.

5.2. Comparative Study

In the first part of the simulation, the LS, ES, DCS, DR mechanisms and the proposed DDCRS are compared in terms of message overhead. The number of sensor nodes is set to 1500 and the duration of each query is 300 s. Figures 11–13 reveal the performance results. In Figure 11, the LS has a smaller number of messages than the ES because only the event data that sinks are interested in will be replied to the sinks periodically, rather than to each data transmitting to the sinks in the ES mechanism. However, as the number of queried event types grows, the LS gets a higher data traffic than the ES because the LS needs to use blind flooding for each query and large number of sensor nodes reply data to sink nodes periodically because all event types are queried. Hence, the total messages of the LS is

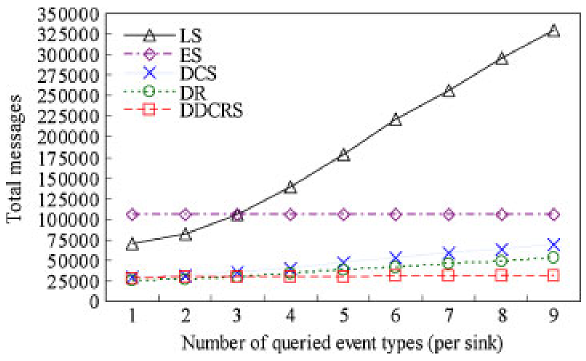


Fig. 11. Performance study of total messages *versus* the number of queried event types.

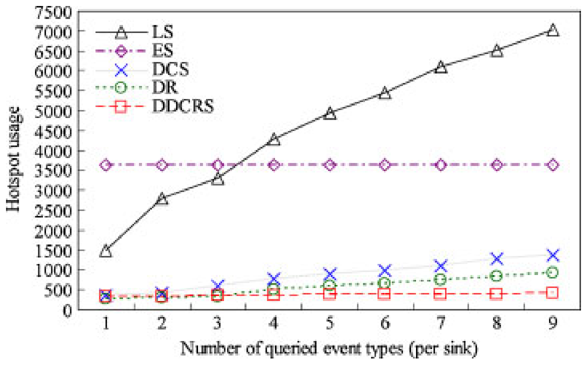


Fig. 12. Performance study of hotspot usage *versus* the number of queried event types.

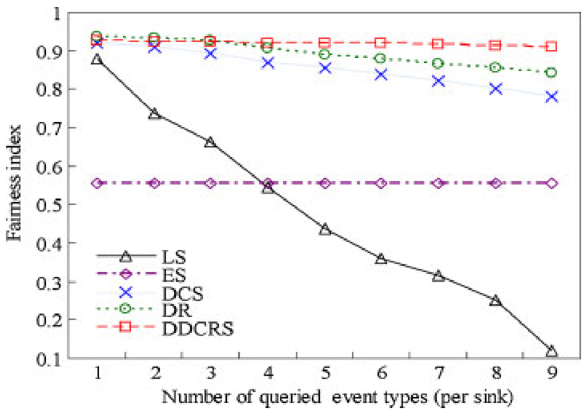


Fig. 13. Performance study of fairness index *versus* the number of queried event types.

increased significantly with the number of queried event types. The total number of messages of the ES is constant because, once sensor nodes detect event data, they send them to sink nodes for storing.

DCS-based mechanisms outperform the LS and ES since they use hash functions to check the location of the data-centric node and then adopts the routing mechanism to send query packets instead of blind flooding. Aside from this, DCS-based mechanisms only reply to interested data from the data-centric node to sink nodes periodically. The number of queried event types therefore has a small impact on message overhead in DCS-based mechanisms. Compared with the DCS mechanism, the DR mechanism provides the distance-sensitive retrieval scheme such that the sink node sends a query to travel along a curve that intersects the replication curve as quickly as possible. When the sink node is close to the sensor node that sends the sensing data to the replication curve, it can find the data quickly. Therefore, the DR mechanism has smaller data traffic compared to the DCS mechanism. When the number of queried event types is small, DDCRS and DCS mechanisms have similar messages overhead since a data-centric node receives more than one query with small probability. However, when the number of queried event types becomes larger than three, the probability that a data-centric node receives more than one queried event type increases. Hence the DDCRS initiates the dynamic DCS mechanism to change data-centric nodes, reducing the total number of messages. In addition, the DDCRS adopts data-centric routing mechanism to construct a shared path for replying data, thereby reducing duplicate messages. As a result, the DDCRS outperforms the other four mechanisms in terms of messages overhead when the number of queried event types is larger than 3.

Hotspot usage [1] represents the link usage of the node with the maximum traffic in the network. A big hotspot usage means that the duration from network start to the end of the first node is short due to power exhaustion. An investigation of relation between the hotspot usage and the number of queried event types is depicted in Figure 12. The ES mechanism has a larger hotspot usage than the LS when number of queried event types is low since the ES mechanism replies to the sink nodes every event data, even though the sink node is not interested in the data. The LS, on the other hand, only replies with the queried data to the sink nodes. However, when the number of queried event types increases, the LS increases the hotspot usage rapidly because of larger number of data sent from the sensor nodes periodically. Since ES replies to the sink nodes every event data, the number of queried event types does not impact the hotspot usage and therefore the hotspot usage of the ES keeps to a constant value as the number of queried event types increases. The hotspot usage of DCS-based mechanisms are not significantly impacted by the number of queried event types. In the DR mechanism, it provides distance-sensitive retrieval scheme such that the traffic load can be well distributed over the network. Therefore, the hotspot usage of the DR mechanism is smaller than that of the DCS mechanism. Compared with the DR mechanism, when the number of queried event types is larger than 3, the DDCRS mechanism reduces the hotspot usage on a data-centric node due to the change of the data-centric node from one sensor to another. Moreover, the DDCRS, applying a shared path mechanism to reply data to multiple sinks, also efficiently reduces hotspot usage due to different routing paths from a data-centric node to multiple sinks.

Figure 13 compares the five mechanisms in terms of fairness index, which represents the degree of power balance. When the number of queried event types is small, the LS has a higher fairness index than the ES. When the number of queried event types is larger than four, the sensor that detects several events with different types will transmit more data to the querying sinks. Hence, the fairness index of LS mechanism is lower than the one of ES mechanism. Since every sensed data has been sent to the sink, the ES is not affected by the number of queried event types. Hence, the fairness index of the ES mechanism keeps to a constant. The DCS-based mechanisms outperform the LS and ES in power balance because they distribute traffic on the data-centric nodes. Compared with the DR and DCS mechanisms, the DDCRS changes data-centric nodes according to the locations and requested frequencies of sink nodes, again distributing traffic over those sensor nodes that have played the role of data-centric node. In addition, the data-centric routing mechanism adopted in the DDCRS also reduces the traffic between data-centric nodes and sink nodes and therefore increases the fairness index indirectly. As a result, the DDCRS outperforms DR and DCS mechanisms in terms of fairness index.

Table II depicts the improvement of the DR and the DDCRS against the DCS by varying the number of queried event types ranging from 4 to 9. The DR and the DDCRS improve the number of total messages by about 16–24% and 25–55%, respectively. In hotspot usage, the DR and the DDCRS improve 33–39% and 54–69%, respectively. As for fairness index, the DR and the DDCRS improve by 4–6% and 9–12%, respectively. The proposed DDCRS mechanism works better in power balance and message overhead due to the change of data-centric location according to sinks' location and requested frequencies. Additionally, the data-centric routing mechanism which constructs shared paths for replying data also reduces the traffic of the hotspot nodes.

Table II. Improvement with different number of queried event types.

	Queries	DCS (%)	DR (%)	DDCRS (%)
Total messages	4	100	116	125
5	100	119	130	
6	100	122	136	
7	100	123	142	
8	100	123	148	
9	100	124	155	
Hotspot usage	4	100	133	154
5	100	134	159	
6	100	135	162	
7	100	135	165	
8	100	137	168	
9	100	139	169	
Fairness index	4	100	104	109
5	100	105	110	
6	100	105	111	
7	100	105	112	
8	100	106	112	
9	100	106	112	

Figures 14–16 evaluate the traffic by varying the network size from 100 to 10 000 sensor nodes, but the node density is fixed to a constant 1/1024 1/m^2. The duration of each query is set by 300 s and the number of event types each sink nodes can query is 4. In Figure 14, the total number of messages of all mechanisms increases along with the network size. The network size impacts on the number of messages significantly in the LS and ES mechanisms because the number of sensor nodes that reply data to sink nodes increases as the network size grows. In DCS-based mechanisms, only few data-centric nodes reply data to sink nodes. The network size impacts the DCS-based mechanism a little due to the increase of route length. When the network size is small, the total numbers of messages of DR and DDCRS mechanisms

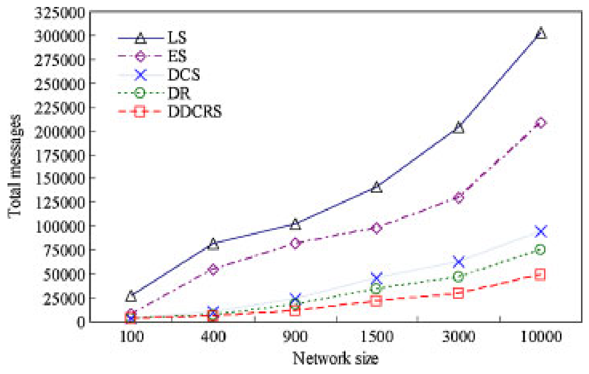


Fig. 14. Performance study of total messages *versus* different network size.

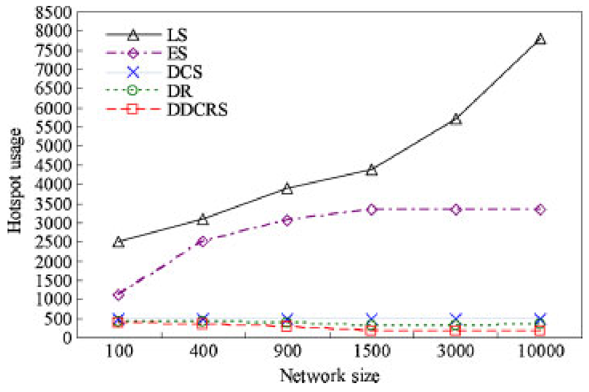


Fig. 15. Performance study of hotspot usage *versus* network size.

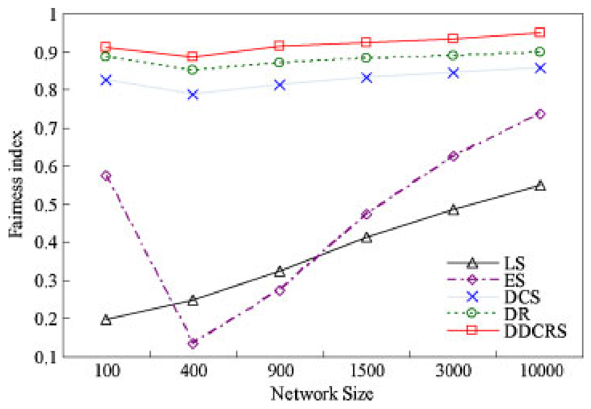


Fig. 16. The impact of network size on the fairness index.

are similar to the result of DCS mechanism. Since the route length is short between the data-centric node and each sink node when the network size is small, the improvement of the DR and DDCRS is limited. However, when the network size grows, DR and DDCRS have significant improvement in terms of total number of messages. This is because the distance-sensitive retrieval scheme proposed in the DR mechanism and the benefit of changing data-centric nodes in the DDCRS mechanism are cost-effective. Hence, the DR and DDCRS work efficiently in a large-scale WSN. Moreover, the DDCRS applying the Data-Centric Routing Mechanism to reduce the duplicated data packets, and therefore it outperforms the DR mechanism in a large-scale network.

Figure 15 depicts the relation of hotspot usage and network size. As the network size increases, the number of sensors that detect the same event type also increases. By applying the LS mechanism, sensors that detect event data which sinks are interested in should reply with information to the sinks, thereby increasing the traffic and hotspot usage significantly. The simulation controls the number of events to make them a constant within a fixed time interval, and each sensor has the same probability of detecting the occurred event. Therefore, when the network size becomes larger than 1500, all events are being detected by the sensors, making the hotspot usage of the ES mechanism a constant. As for DCS-based mechanisms, the hotspot usage is kept low even though the network size grows. Compared with the DCS mechanism, the DR mechanism applying the distance-sensitive retrieval scheme can efficiently balance the traffic load in the sensor network. Therefore, the hotspot usage of the DR mechanism is smaller than that of the DCS mechanism. In the DDCRS mechanism, the change of the data-centric nodes and construction of shared paths significantly reduce the hotspot usage. As a result, the DDCRS outperforms the DR in terms of hotspot usage.

Figure 16 compares the performance of all mechanisms with various network sizes in terms of fairness index. The network traffic is uniformly distributed over the whole WSN, and each sensor node has the same probability of detecting an event. When the network size becomes smaller than 400, the fairness index of the ES and DCS-based mechanisms decreases with the network size. This is because sensor nodes that are close to sink nodes or data-centric nodes have higher traffic than nodes at other locations. Although the fairness index of DCS-based mechanisms also decreases, it is not significant because the traffics for storing data are distributed to several data-centric nodes. In the LS mechanism, since the detected event data are stored in a sensor's local memory without creating traffic, the traffic of the LS highly depends on the query of sink nodes, and therefore the fairness of the LS is low. When the network size is larger than 400, the fairness index of the ES increases significantly because the number of detected events is fixed, and the detected events are distributed over the entire WSN. As the network size grows to 1500, the ES has a higher value of fairness index than the LS. Since the network traffic is totally distributed over the data-centric nodes, DCS-based mechanisms obtain a higher fairness index than the LS and the ES. In addition, the DDCRS outperforms the other DCS-based mechanisms in terms of fairness index because it changes data-centric nodes dynamically and constructs the shared paths.

Table III lists the improvement of the DR and the DDCRS against the classic DCS by varying network sizes ranging from 1500 to 10 000. The DR and the DDCRS improve by 20–25% and 48–54%, in terms of the total messages respectively. Compared to the DCS, the DR and the DDCRS improve 33–34% and 62–66% in terms of hotspot usage, respectively. Furthermore,

Table III. Improvement with different network size.

	Network size	DCS (%)	DR (%)	DDCRS (%)
Total messages	1500	100	120	148
3000	100	122	151	
10 000	100	125	154	
Hotspot usage	1500	100	133	162
3000	100	133	163	
10 000	100	134	166	
Fairness index	1500	100	103	106
3000	100	103	106	
10 000	100	104	107	

the DR and the DDCRS respectively improve by 3–4% and 6–7% in terms of fairness index. Since each sensor has a constant probability of event detection, the total number of events increases as the network size grows. The total messages and hotspot usage of the DCS therefore increase with the network size. However, the DDCRS changes the locations of data-centric nodes according to the sinks' locations and the requested data collection frequencies. Therefore, the benefit obtained from the change of the data-centric node is more significant as network size grows, resulting in the DDCRS with a better performance compared to the DCS in terms of total messages, hotspot usage and fairness index. Furthermore, DDCRS constructs a shared path to reply data, additionally improving the performance of the DCS in terms of total messages, hotspot usage and fairness index.

Figures 17–19 compare the performance of different mechanisms, while the duration of each query varies, ranging from 50 to 400 s. The network size is set to 1500 sensor nodes, and the number of event types is set to four. As shown in Figure 17, the number of total messages of the ES is kept constant because all of the event data have been stored in sink nodes and it does not create traffic for the sink's queries. The total number of messages of the LS and the DCS-based mechanisms increase with the duration of queries. The duration time has a significant impact on traffic in the LS because the event data are stored in many sensor nodes, resulting in a larger number of data packets replying to sinks periodically. The DCS-based mechanisms store the data in data-centric nodes and therefore the duration time only affects the traffic on data-centric nodes. As a result, the duration of each query minimally affects DCS-based mechanisms. When the query duration is controlled at 50 s, data-centric nodes have a low probability to be requested by more than two queries. When a data-centric node receives a query from another sink node, the old query would be almost overdue. Therefore, DCS and DDCRS mechanisms have similar performance results when the query duration is smaller than 50 s. Compared with the other two DCS-based mechanisms, the DR mechanism can reduce efficiently the total number of messages by applying the distance-sensitive retrieval scheme and therefore has better performance in terms of the total number of messages. However, when the duration time increases, data-centric nodes may have to reply to two or more sink nodes at the same time. Since the DDCRS dynamically changes data-centric nodes according to the benefit evaluation in dynamic DCS mechanism, it can efficiently reduce the total number of messages in the WSN. Moreover, the DDCRS uses shared paths to reduce traffic from data-centric nodes to multiple sink nodes, resulting in less traffic in event data delivery. Hence, the DDCRS outperforms DR and DCS mechanisms and works well in applications that demand collecting data for a long period of time.

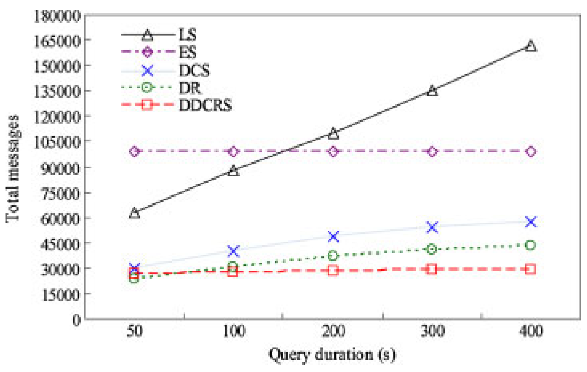


Fig. 17. The impact of query duration on message overhead.

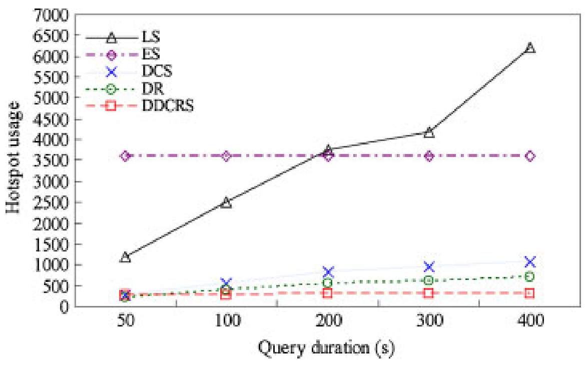


Fig. 18. The impact of query duration on hotspot usage.

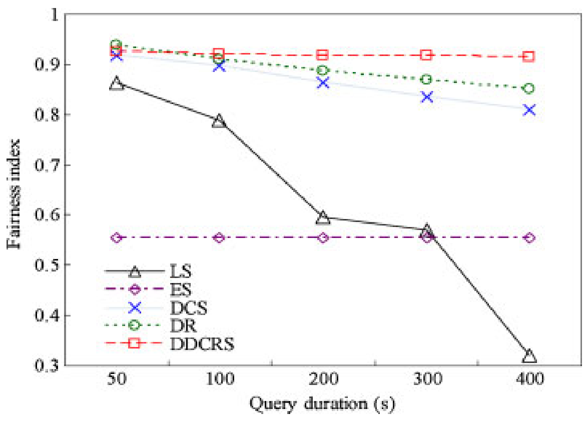


Fig. 19. The comparison of fairness indices of the five compared mechanisms by varying the query duration.

Figure 18 examines the impact of query duration on hotspot usage. The LS and DCS-based mechanisms increase with the query duration, but the ES keeps a constant. Since the DCS mechanism fixes data-centric nodes, the hotspot usage increases significantly. In the DR mechanism, the distance-sensitive retrieval scheme can balance the traffic load of the data-centric nodes. Therefore, the hotspot usage of the DR mechanism is smaller than that of the DCS mechanism. Compared to the DR mechanism, the DDCRS mechanism distributes traffic of data-centric nodes on HD node, ND node and several OD nodes. Therefore, even though the query duration is long, the proposed DDCRS mechanism does not significantly increase the hotspot usage. Moreover, the DDCRS has the lowest hotspot usage because it reduces duplicated transmissions of event data from data-centric nodes to multiple sinks by constructing a shared routing path.

Figure 19 investigates the impact of query duration on energy balance. In Figure 19, the fairness index of the ES mechanism keeps to a constant since sensor nodes reply to each detected event information to sink irrelative to the sink's query. In the LS, the number of messages that sensor nodes reply to sink nodes depends on the number and the duration of the queries requested from the sink nodes. Therefore, the fairness index of the LS decreases with the query duration. In the DCS, since data-centric nodes are fixed without change, the traffic on data-centric nodes increases with the query duration. Hence, the fairness index of the DCS decreases with the query duration. The DR mechanism outperforms DCS mechanism in terms of fairness index because the applied distance-sensitive retrieval scheme efficiently distributes the traffic load over the network. Although the fairness index of the DDCRS is affected by query duration, the impact is not significant because data-centric nodes change one another. Moreover, the DDCRS adopts shared paths to reduce duplicated traffic on data-centric nodes, additionally improving the fairness factor. As a result, the DDCRS outperforms the other mechanisms in terms of fairness index.

Table IV lists the improved results of the DR and the DDCRS against the classic DCS in varying query duration ranging from 100 to 400 s. In total messages, the DR and the DDCRS improve by 23–26% and 31–50%, respectively. In hotspot usage, the DR and the DDCRS improve by 25–35% and 48–69%, respectively. Furthermore, in fairness index, the DR and the DDCRS improve by 2–6% and 3–12%, respectively. The number of replied data increases with the query duration. The total messages and hotspot usage of DCS therefore increase, while the fairness index of the DCS decrease. However, the DDCRS distributes the traffic of data-centric nodes over the HD node, the ND node and several the OD nodes. The DDCRS therefore reduces hotspot usage and increases fairness index significantly. Moreover, the DDCRS constructs a shared path and reduces redundant data traffic, resulting in more significant improvement as the query duration increases.

Table IV. Improvement in different query duration.

	Query duration	DCS (%)	DR (%)	DDCRS (%)
Total messages	100	100	123	131
200	100	124	142	
300	100	124	147	
400	100	126	150	
Hotspot usage	100	100	125	148
200	100	128	152	
300	100	130	160	
400	100	135	169	
Fairness index	100	100	102	103
200	100	103	105	
300	100	105	109	
400	100	106	112	

5.3. Impact of Angle Threshold, Data Collection Frequency and Event Producing Frequency

The benefit evaluation of the proposed DDCRS mechanism uses an angle threshold α to predict whether or not two sink nodes can benefit from a shared path. In Figures 20–22, the threshold α varies by 45°, 90°, 135°, and 180° for comparison and the network size is controlled by 1500 nodes. To increase the opportunity of shared path among sink nodes, each query's duration is set to 400 s. The number of event types that sink nodes can query varies from one up to 9.

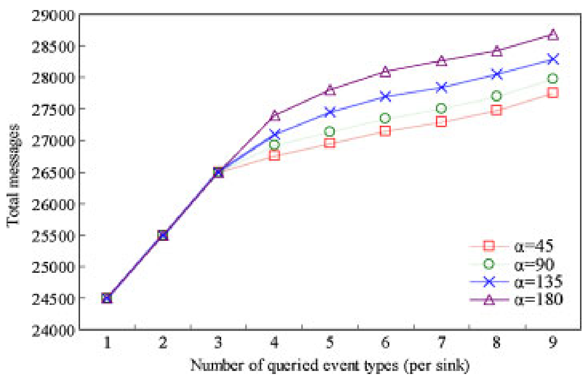


Fig. 20. The impact of angle α on message overhead.

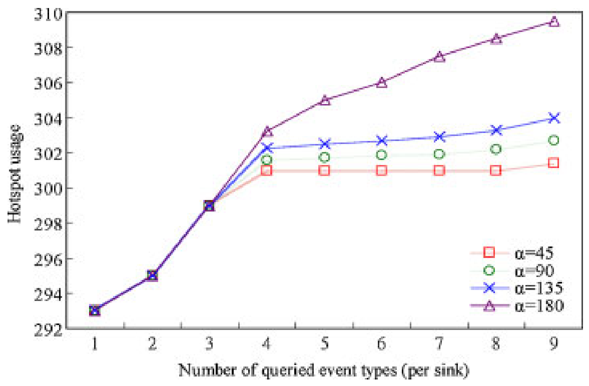


Fig. 21. The impact of angle α on hotspot usage.

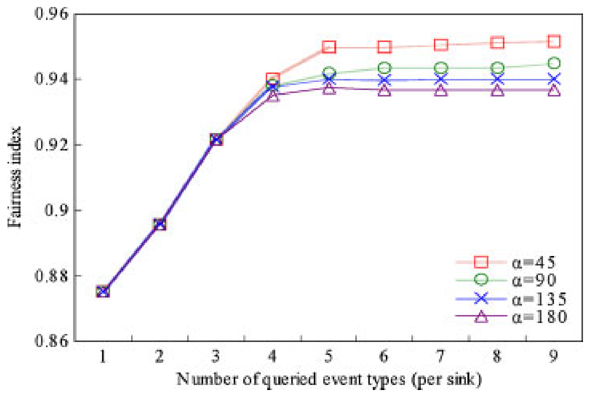


Fig. 22. The impact of angle α on fairness index.

In Figure 20, when the number of queries event types is small, the probability that a data-centric node receives a query from more than two sink nodes is low. However, the designed *data-centric node change procedure* is initiated when more than two sink nodes' query arrive at the data-centric node. Therefore, the developed mechanism will not be initiated, resulting in similar performance results. Once the number of queries becomes larger than 3, the designed DDCRS mechanism is initiated. The *benefit evaluation procedure* is therefore initiated to evaluate whether or not the data-centric node should be changed. However, the opportunities of constructing a shared path depend on the threshold α. In case that α is set by a larger value, two sinks with large angle will be included in the share group. Hence, a short segment shared by the two sinks will be constructed. Since the number of reduced messages is increased with the length of shared segment, the construction of short segment will save few messages but create additional control overheads for changing the data centric node, resulting in a poor performance in terms of the total number of messages. As Figure 20 shows, the number of total messages is large in case α is 180°.

Figure 21 also reveals that a large angle threshold increases the opportunities for changing data-centric node due to the inaccuracy of benefit evaluation, making hotspot usage impossible to be efficiently reduced. On the other hand, a small value of angle threshold increases the accuracy of benefit evaluation; therefore, DDCRS changes the data-centric node efficiently.

Figure 22 depicts the impact of angle threshold value on the fairness index. Similarly, a large value of α also has a small fairness index due to the inaccuracy of benefit evaluation. The DDCRS mechanism makes a decision that the data-centric node is worthwhile to change as the angle threshold is set by a large value. A small value of α has high accuracy of benefit evaluation; therefore, DDCRS changes data-centric nodes efficiently, and thereby increasing the fairness index.

The proposed mechanism changes DCS location to a better place in monitoring areas according to query frequencies from multiple sinks' requests. Figures 23–25 investigate the improvement of the DDCS and the DDCRS against the DCS in varying the standard deviation (STDEV) of query frequencies, ranging from 0 to 105. Here, the number of event types is set to four, network size is set to 1500 and the query duration is set to a constant value of 300 s. In Figure 23, the improvement of the DDCS and the DDCRS in total messages increases with the standard deviation of query frequencies. As the difference of query frequencies becomes significant, the two mechanisms change the data-centric node toward the sink node with the largest query frequency. The route length

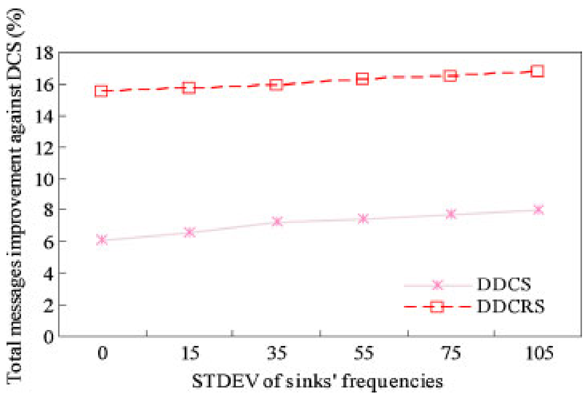


Fig. 23. The comparison of DDCS and DDCRS in terms of the number of total messages by varying the STDEV of sinks' frequencies.

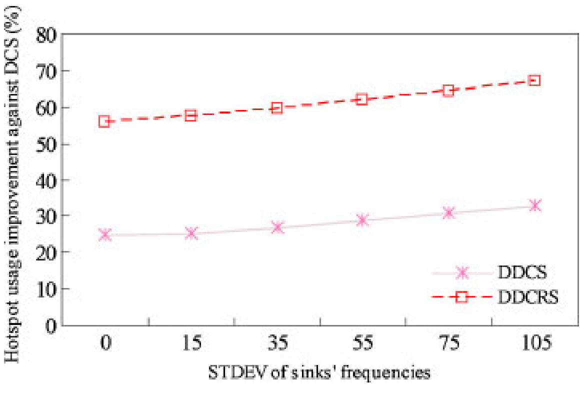


Fig. 24. The comparison of DDCS and DDCRS in terms of hotsopt usage by varying the STDEV of sinks' frequencies.

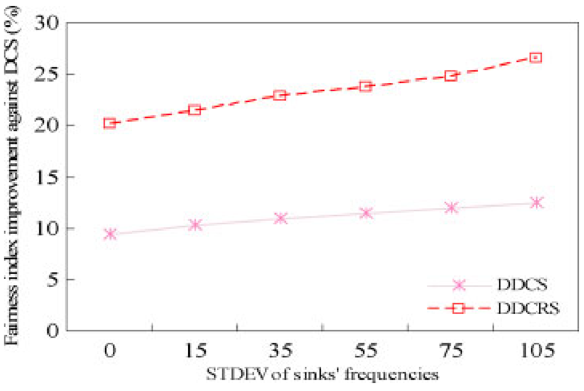


Fig. 25. The comparison of DDCS and DDCRS in terms of fairness index of power consumption by varying the STDEV of sinks' frequencies.

from data centric node to the sink node with largest query frequency can therefore be reduced, reducing the number of messages and the hotspot usage and increasing fairness index, as shown in Figures 23–25. Moreover, the DDCRS mechanism constructs shared paths by selecting the neighbor which has maximal sum of frequencies to be the next forwarding node. Since the route length between the data-centric node and the sink node with larger frequency is decreased, the total number of transmitted data packets can be reduced. As a result, the DDCRS mechanism outperforms the DDCS mechanism in terms of total messages, hotspot usage and fairness index.

Figure 26 compares the DCS and DDCRS mechanisms in terms of the energy consumption by varying the event producing frequency and STDEV of query frequencies. The number of event types is set to 4. The network size is set to 1500 and the query duration is set to a constant value of 300 s. The energy

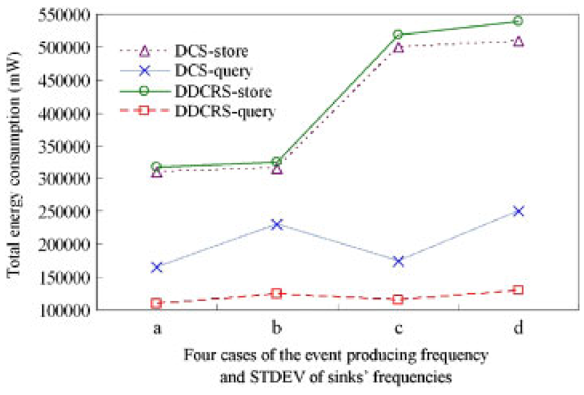


Fig. 26. The impact of the event producing frequency and STDEV of sinks' frequencies on the total energy consumption.

consumptions for transmitting and receiving a packet are 14.88 and 12.50 mW [21], respectively. Here, the types of event producing frequency and STDEV of query frequencies are divided into four cases: (a) 15 and 15 (b) 15 and 75 (c) 75 and 15 (d) 75 and 75. As shown in Figure 26, the total energy consumption of the DCS-store and DDCRS-store is increased with the event producing frequency. This is because that more sensing data will be sent to the data-centric nodes when the event producing frequency grows. Moreover, the DDCRS-store is worse than that of the DCS-store due to the additional storage overhead raised by changing the data-centric nodes in the DDCRS mechanism. On the other hand, the total energy consumptions of DCS-query and DDCRS-query are also increased with the STDEV of sinks' frequencies. However, the DDCRS mechanism can significantly improve the total energy consumption due to the change of data-centric nodes and the construction of shared paths. As a result, the DDCRS-query outperforms the DCS-query in terms of the total energy consumption.

6. Conclusion

This paper has proposed a novel DDCRS. The developed routing mechanism automatically constructs shared paths from data-centric nodes to multiple sinks, reducing duplicate packets transmission, and therefore saving the energy consumption of forwarding nodes. In addition, a dynamic DCS mechanism has also been proposed to determine the better location for the new data-centric node. A benefit evaluation procedure has been developed to estimate the benefit and the overhead of changing the data-centric node, ensuring that this change is cost-effective. The simulation results show that the DDCRS outperforms existing data storage mechanisms in message overhead, power consumption, and power balancing for applications of long time data collection with a large-scale WSN.

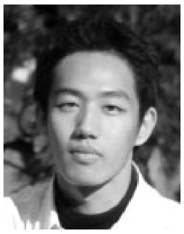
References

1. Ratnasamy S, Karp B, Yin L, *et al*. Data-centric storage in sensornets with GHT, a geographic hash table. *Journal on Mobile Networks and Applications*, 2003; **8**(4): 427–442.
2. Ratnasamy S, Karp B, Yin L, *et al*. GHT: a geographic hash table for data-centric storage. In *First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, GA, USA, September 2002.
3. Le TN, Yu W, Bai X, Xuan D. A dynamic geographic hash table for data-centric storage in sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2006; 2168–2174.
4. Karp B, Kung HT. GPSR:greedy perimeter stateless routing for wireless networks. In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, August 2000; 243–254.
5. Newsome J, Song D. GEM: graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of international conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, November 2003; 76–88.
6. Seada K, Helmy A. Rendezvous regions: a scalable architecture for service location and data-centric storage in large-scale wireless networks. In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, San Diego, CA, September 2003.
7. Zhang W, Cao G, Porta TL. Data dissemination with ring-based index for wireless sensor networks. *IEEE Transactions on Mobile Computing* 2007; **6**(7): 832–847.
8. Fang Q, Gao J, Guibas LJ. Landmark-based information storage and retrieval in sensor networks. In *Proceedings of the 25th Conference of the IEEE Communication Society (INFOCOM)*, April 2006.
9. Fang Q, Gao J, Guibas L, de Silva V, Zhang L. GLIDER: gradient landmark-based distributed routing for sensor networks. In *Proceeding of the 24th Conference of the IEEE Communication Society (INFOCOM)*, March 2005.
10. Sarkar R, Zhu X, Gao. J. Double rulings for information brokeragein sensor networks. In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, Los Angeles, CA, USA, September 2006; 286–297.
11. Navstar GPS Operation. [Online]. Available: tycho.usno.navy.mil/gpsinfo.html
12. He T, Huang C, Blum BM, Stankovic JA, Adbelzaher T. Range-free localization schemes for large scale sensor networks.In *Proceedings of International Conference on Mobile Computing and Networking (MobiCom)*, San Diego, CA, USA, September 2003; 81–95.
13. Bulusu N, Heidemann J, Estrin D. GPS-less low cost outdoor location for very small devices. *IEEE Personal Communications Magazine* 2000; **7**(5): 28–34. Special Issue on 'Smart Space and Environments'.
14. Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H. Chord: a scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, August 2001.
15. Rowstron A, Druschel P. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systmes. *Journal of Lecture Notes in Computer Science* 2001; **2218**: (329).
16. Zhao B, Kubiatowicz J, Joseph A. Tapestry: an infrastructure for faul-tolerant wide-area location and routing. *Technical Report UCB/CSD-01-1141*, Computer Science Division, University of California at Berkeley, Berkeley, CA, April 2001.
17. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S. A scalable content-addressable network, ACM SIGCOMM Computer Commnication Review. In *Proceedings of Internationa Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 31, no. 4, San Diego, CA, August 2001; 161–172.
18. Melkonian V. New primal-dual algorithms for Steiner tree problems. *Computers and Operations Research*, 2007; **34**(7): 2147–2167.
19. Zeng X, Bagrodia R, Gerla M. GloMoSim: a library for parallel simulation of large-scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*, Canada, 1998; 154–161.
20. Jain R. The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley and Sons Inc.: New York, NY, 1991; 36–387.
21. Sabbineni H, Chakrabarty K. Location-aided flooding: an energy-efficient data dissemination protocol for wireless sensor networks. *IEEE Transactions on Computers*, 2005; **54**(1): 36–46.

Authors' Biographies

Chih-Yung Chang received the Ph.D. in Computer Science and Information Engineering from National Central University, Taiwan, in 1995. He joined the faculty of the Department of Computer and Information Science at Aletheia University, Taiwan, as an Assistant Professor in 1997. He was the Chair of the Department of Computer and Information Science, Aletheia University, from August 2000 to July 2002. Since August 2002, he has been an Associate Professor with the Department of CSIE, Tamkang University. He is currently a Full Professor with the Department of CSIE at Tamkang University, Taiwan. Dr. Chang served as an Associate Guest Editor of several International Journals, including *Journal of Telecommunication Systems*(2009), *Journal of Information Science and Engineering*(*JISE*, 2008), *Journal of Internet Technology*(2004 and 2008), *Journal of Mobile Multimedia* (2005). He also served as a member of Editorial Board of *Tamsui Oxford Journal of Mathematical Sciences* (2001–2008) and *Journal of Information Technology and Applications* (JITA, 2008). He was an Area Chair of *IEEE AINA*'2005, Vice Chair of *IEEE WisCom*'2005 and *EUC*'2005, Track Chair (Learning Technology in Education Track) of *IEEE ITRE*'2005, Program Co-Chair of *WASN*'2007, *MNSAT*'2005 and *UbiLearn*'2006, Workshop Co-Chair of *ICS*'2008, *INA*'2005, *MSEAT*'2003, *MSEAT*'2004, and Publication Chair of *MSEAT*'2005 and *SCORM*'2006. Dr. Chang is a member of the IEEE Computer Society, Communication Society and IEICE society. His current research interests include wireless sensor networks, bluetooth radio networks, ad hoc wireless networks, and WiMAX broadband technologies.

Jang-Ping Sheu received the B.S. degree in computer science from Tamkang University, Taiwan, R.O.C., in 1981 and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Taiwan, in 1983 and 1987, respectively. He was the chair of the Department of Computer Science and Information Engineering, National Central University from 1997 to 1999. He was the director of the Computer Center, National Central University, from 2003 to 2006. He is currently a chair professor of the Department of Computer Science, National Tsing Hua University. His current research interests include wireless communications and mobile computing. He was an associate editor for the *Journal of the Chinese Institute of Electrical Engineering*, the *Journal of Information Science and Engineering*, the *Journal of the Chinese Institute of Engineers*, and the *Journal of Internet Technology*. He is an associate editor for the *IEEE Transactions on Parallel and Distributed Systems*, the *International Journal of Ad Hoc and Ubiquitous Computing*, and *the International Journal of Sensor Networks*. He received the Distinguished Research Awards from the National Science Council of the Republic of China in 1993–1994, 1995–1996, and 1997–1998, the Distinguished Engineering Professor Award from the Chinese Institute of Engineers in 2003, the certificate of Distinguished Professorship from the National Central University in 2005, and the K.-T. Li Research Breakthrough Award from the Institute of Information and Computing Machinery in 2007. He is a fellow of the IEEE and a member of the ACM and the Phi Tau Phi Society.



Sheng-Wen Chang received the B.S. degree in Computer Science and Information Engineering from Tamkang University, Taiwan, in 2004. Since 2005, he was working toward his Ph.D. degree and currently he is a Ph.D. candidate in Department of Computer Science and Information Engineering at Tamkang University. Mr. Chang is a student member of the IEEE Computer Society, Communication Society and IEICE society. He has won lots of scholarships in Taiwan and participated in many WiMAX, bluetooth radio networks, wireless sensor networks, and ad hoc wireless networks projects. His research interests include LTE, WiMAX, bluetooth radio networks, wireless sensor networks and ad hoc wireless networks.

Yu-Chieh Chen received the B.S. degree in Computer Science and Information Engineering from Ming Chuan University, Taiwan, in 2005, and the M.S. degree in Computer Science and Information Engineering from Tamkang University, Taiwan, in 2007. Since 2007, he was working toward his Ph.D. degree in Department of Computer Science and Information Engineering at Tamkang University. Mr. Chen won lots of scholarships in Taiwan and participated in many Wireless Sensor Networking projects. His research domains include wireless sensor networks, ad-hoc wireless networks, mobile/wireless computing, and WiMAX.