A Data-Centric Mechanism for Wireless Sensor Networks with Weighted Queries

Li-Ling Hung[+], Sheng-Wen Chang[*], Chih-Yung Chang[*], Yu-Chieh Chen[*]
*Tamkang University, Tamsui, Taipei, Taiwan,
[+]Aletheia University, Tamsui, Taipei, Taiwan
llhung@mail.au.edu.tw, swchang@wireless.cs.tku.edu.tw, cychang@mail.tku.edu.tw, ycchen@wireless.cs.tku.edu.tw

Abstract*—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 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 node 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.

I. INTRODUCTION

A 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 local storage mechanisms, data is 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. External storage, on the other hand, proposes another alternative mechanism. Once a sensor node detects an event, the data is 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 data-centric storage 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 is 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 study [1] has proposed an efficient Data-Centric Storage 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 [2] routing protocol. When a sink node intends to collect the information of an event, it uses the hash function of the GHT to obtain the location where the event is stored and then adopts a GPSR 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.

An index-based architecture [3] 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. In study [4], 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 articles devoted themselves to develop data-centric storage 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 Data-Centric Storage mechanisms for a multi-sink environment. Firstly, a dynamic data-centric storage 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 remainder of the paper is organized as follows. In Section 2, a multi-sink network environment is described. Section 3 presents an overview of the developed mechanisms, and illustrates the Data-Centric Routing Mechanism and Dynamic Data-Centric Storage Mechanism in detail. Performance study is presented in Section 4 and finally, Section 5 concludes the paper.

II. NETWORK ENVIRONMENT

The network model is similar to previous works [1, 3] 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.

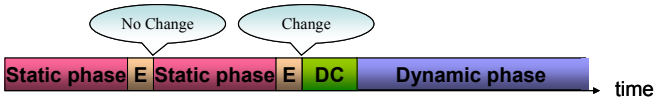
978-1-4244-5228-6/09/$26.00 ©2009 IEEE

Each sensor node is aware of its own location 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. 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.

III. DYNAMIC DATA-CENTRIC ROUTING AND STORAGE MECHANISMS

3.1. Protocol Architecture Overview

The proposed Dynamic Data-Centric Storage Mechanism (DDCRS) 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 [1]. 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 [2]. Once the location of the data-centric node changes, the Dynamic Data-Centric Storage Mechanism will switch to a dynamic phase, handling both the data storage and the delivery problems.



E: Benefit Evaluation (Benefit v.s Overhead)
DC: Data-centric node change procedure

Figure 1: Static and dynamic phases of the proposed data centric storage mechanism.

As shown in Fig. 1, 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 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 Data-Centric Storage 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. 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.

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 [5], which aims to minimize the sum of the weighted Euclidean distances. Since the generalized Steiner tree problem is an NP-hard problem [5], 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.

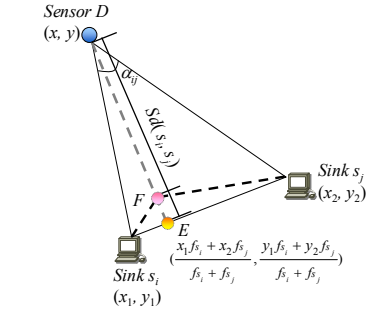
For ease of description, some symbols are defined below. Let $d(A, B)$ denote the distance between nodes A and B. Let $ShareGroup(s_1, s_2, \ldots, s_k)$ represent that k sinks $s_1, s_2, \ldots, s_k$ can share the same path. Assume there are n sinks $s_1, s_2, \ldots, s_n$ that request data from data-centric node D. Let fs_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 $n_1, n_2, \ldots, n_m$. As shown in Fig. 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 $w_i = \sum_{s_j \in SinkNodeSet(n_i)} f_{s_j}$. Let p be the current forwarding node and p has m neighbors $n_1, \ldots, n_m$. Node p will select neighbor n_x to be the next forwarding node if $w_x \geq w_i$ for all $1 \leq i \leq m$. That is, the next forwarding node n_x should satisfy the condition $w_x = \max(w_1, w_2, \ldots, w_m)$. 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.

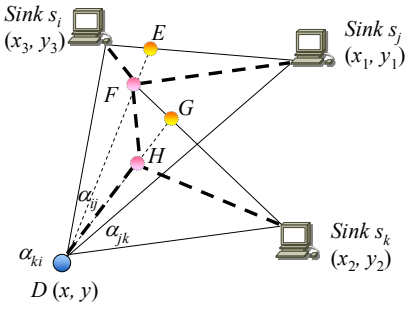
Initial:
ReplySink = ∅
SelectedNeighbor = ∅
Begin
while |*ReplySink*| < *n*
select n_x to be the next forwarding node,
where n_x satisfies $w_x = \max(w_1, w_2, \ldots, w_m)$ and $1 \le x \le m$
insert *SinkNodeSet*(n_x) into *ReplySink* set
insert n_x into *SelectedNeighbor* set
remove *SinkNodeSet*(n_x) from NIT
end while
Construct routing table with *SelectedNeighbor* set
End

Some other complicated case may occur since it is possible that two neighbors can forward to the same sink at the same time. Therefore, two results of shared routes are possible. 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.

$$Sd(s_i, s_j) = \begin{cases} 1 & , \text{if} \quad \alpha_{ij} = 0° \\ 1 - \dfrac{\alpha_{ij}}{180°} & , \text{if} \quad 0 < \alpha_{ij} < 180° \\ 0 & , \text{if} \quad \alpha_{ij} = 180° \end{cases} \tag{1}$$



(a)



(b)

Figure 2: 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 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 $fs_i > fs_j$. As shown in Fig. 2(a), a median point E with coordination $(\frac{x_1 fs_i + x_2 fs_j}{fs_i + fs_j}, \frac{y_1 fsi + y_2 fs_j}{fs_i + fs_j})$ 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 $Sd(s_i, s_j)$ and point E below. First, $d(D, E)$ and vector $\overrightarrow{DE}$ are calculated using Equations (2) and (3).

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

$$\overrightarrow{DE} = (\frac{x_1 fs_i + x_2 fs_j}{fs_i + fs_j} - x, \frac{y_1 fs_i + y_2 fs_j}{fs_i + fs_j} - y) \tag{3}$$

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

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

Hereafter, $d(D, F)$ can be calculated by shared degree $Sd(s_i, s_j)$ and unit vector $\vec{u}$, as shown in Equation (5)

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

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

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

The point F is a branch point of shared path of s_i and s_j. Given a query frequency fs_i and fs_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 $Max(fs_i, fs_j)$ 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 fs_i and fs_j, respectively. The cost of a shared path of sinks s_i and s_j can therefore be measured by Equation (7).

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

In case there are more than two sinks, say s_1, s_2, …, 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 Fig. 2(b) depicts, sinks s_i, s_j, and s_k share the same data packet. Suppose that $fs_i > fs_j > fs_k$. 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 Fig. 2(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:

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

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 $Max(fs_i, fs_j, fs_k)$ and $Max(fs_i, fs_j)$ 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 Data-Centric Storage Mechanism that dynamically determines a better location of a data-centric node according to sink nodes' location and requested data collection frequency. 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, 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, …, 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:

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

The *OD* node then estimates the benefit and the overhead of changing the data-centric node. The benefit evaluation will be explained in detail in the next subsection. In case it is worthwhile to change the data-centric node, the *OD* node sends an *alert* packet to find *ND* which is closest to the location (x, y) by using the GPSR 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.

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.

(1) New sink query— In the dynamic phase, sink nodes that have never queried the event before are not aware that the 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.

(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, …, 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 can not 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 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.

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.

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, …, s_n are t_1, t_2,…, t_n, respectively, and $t_1 > t_2 > \ldots > t_{n-1}$. In case $t_n > t_{n-1} > t_{threshold}$, it shows that the shortest remaining duration is long enough for the change of the data-centric node, where $t_{threshold}$ is a threshold value of remaining duration of the query. On the other hand, if $t_{n-1} < t_{threshold}$, 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 $t_{n-1} > t_n > t_{threshold}$, 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:
Let $t_{min}=min\ (t_1, t_2, \ldots, t_n)$.
If ($t_{min} > t_{threshold}$) /* 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.

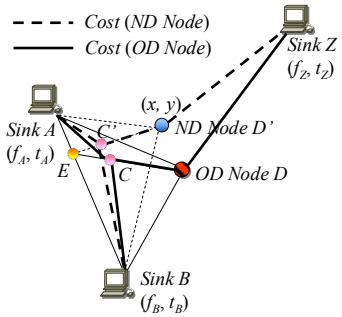


Figure 3: An example for illustrating benefit evaluation.

Take Fig. 3 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 Fig. 8. In case the 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 is the total data size of the event data stored in the *OD* node and sinks' information.

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

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.

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

, 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 Fig. 2.

IV. SIMULATION

This section investigates the performance of the proposed Dynamic Data-Centric Routing and Storage Mechanisms (DDCRS). The following first describes the simulation environment then shows the investigated simulation results.

The proposed DDCRS mechanism was implemented in GloMoSim (version 2.03) [6] and is compared with four storage mechanisms: LS, ES, DCS [1] and Double Ruling [4] (DR in short). In the DDCRS mechanisms, a data-centric node considers executing the Dynamic Data-Centric Storage 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, and DDCRS mechanisms belong to DCS-based mechanisms. The number of sensor nodes is set to 1500. 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 1/10dps, 1/20dps, and 1/40dps (data per second), respectively. The query generation rate of each sink node is 1/10qps (query per second). That is, each sink node sends a query every 10s. 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. 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. Each result is obtained from an average of 10 experiments. The 95% confidence interval is always smaller than ±5% of the reported values.

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

Figure 4 compares the five mechanisms in terms of total message by varying the number of queried event types. The duration of each query is 300s. In Fig. 4, the total messages of the LS is increased significantly with the number of queried event types since the LS needs to use blind flooding for each query and large number of sensor nodes reply data to sink nodes periodically. The total number of messages of the ES is constant since sensor nodes send all detected event data, 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 Data-Centric Storage 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.

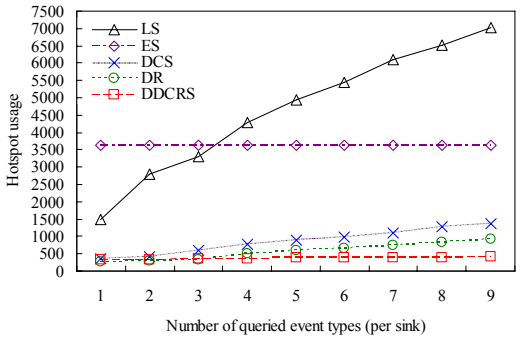


Figure 4: The comparison of the five mechanisms in terms of total messages by varying the number of queried event types.

Figure 5 compares the performance of different mechanisms, while the duration of each query varies, ranging from 50s to 400s. The number of event types is set to four. The total number of messages of the LS and the DCS-based mechanisms increase with the duration of queries. 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. Since the DDCRS dynamically changes data-centric nodes according to the benefit evaluation in Dynamic Data-Centric Storage 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.

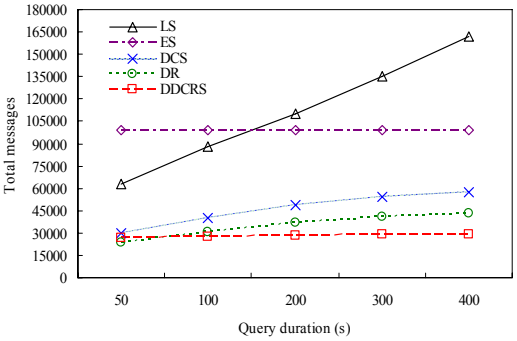


Figure 5: The impact of query duration on message overhead.

Figure 6 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.

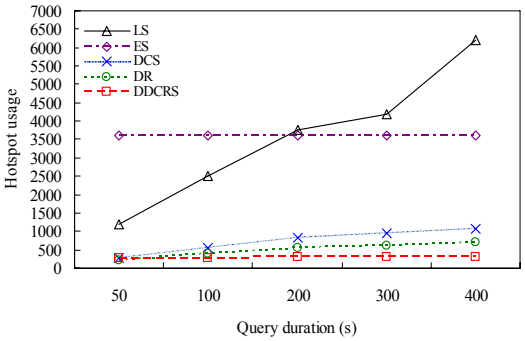


Figure 6: The impact of query duration on hotspot usage.

V. CONCLUSION

This paper has proposed a novel Dynamic Data-centric Routing and Storage Mechanisms. 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 data-centric storage 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] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table," *Journal on Mobile Networks and Applications*, vol. 8, no. 4, August 2003, pp. 427-442.

[2] B. Karp and H. T. Kung, "GPSR:Greedy Perimeter Stateless Routing for Wireless Networks," *Proceedings of International Conference on Mobile Computing and Networking* (MobiCom), pp. 243-254, Boston, Massachusetts, August 2000.

[3] W. Zhang, G. Cao, and T. L. Porta, "Data Dissemination with Ring-Based Index for Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 7, July 2007, pp. 832-847.

[4] R. Sarkar, X. Zhu, and J. Gao, "Double Rulings for Information Brokeragein Sensor Networks," *Proceedings of International Conference on Mobile Computing and Networking* (MobiCom), pp. 286-297, Los Angeles, California, USA, September 2006.

[5] Vardges Melkonian," New primal-dual algorithms for Steiner tree problems," *Computers and Operations Research*, vol. 34, no. 7, July 2007, pp. 2147-2167.

[6] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a Library for Parallel Simulation of Large-Scale Wireless Networks" *Proceedings of the 12th Workshop on Parallel and Distributed Simulations*, pp. 154 - 161, Canada, 1998.