Computer Communications 33 (2010) 92–102

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Achieving optimal data storage position in wireless sensor networks

Zhaochun Yu [a,b,c], Bin Xiao [b,*], Shuigeng Zhou [a]

[a] Department of Computer Science and Engineering, Fudan University, China
[b] Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
[c] Data Center (Shanghai), Industrial and Commercial Bank of China, China

ARTICLE INFO

Article history:
Received 27 April 2008
Received in revised form 20 May 2009
Accepted 5 August 2009
Available online 15 August 2009

Keywords:
Wireless sensor networks
Optimal data storage
Data rate
Geographical location

ABSTRACT

Data storage in wireless sensor networks (WSNs) involves *producers* (such as sensor nodes) storing in storage positions a large amount of data which they have collected and *consumers* (e.g., base stations, users, and sensor nodes) then retrieving that data. When addressing this issue, previous work failed to utilize data rates and locations of multiple producers and consumers to determine optimal data storage positions to be communication cost-effective in a mesh network topology. In this paper, we first formalize the data storage problem into a *one*-to-*one* (*one* producer and *one* consumer) model and a *many*-to-*many* (*m* producers and *n* consumers) model with the goal of minimizing the total energy cost. Based on above models, we propose optimal data storage (ODS) algorithms that can produce global optimal data storage position in linear, grid, and mesh network topologies. To reduce the computation of ODS in the mesh network topology, we present a near-optimal data storage (NDS) algorithm, which is an approximation algorithm and can obtain a local optimal position. Both ODS and NDS are locality-aware and are able to adjust the storage position adaptively to minimize energy consumption. Simulation results show that NDS not only provides substantial cost benefit over centralized data storage (CDS) and geographic hash table (GHT), but performs as well as ODS in over 75% cases.

1. Introduction

Data storage in wireless sensor networks (WSNs) [3,14] involves producers (such as sensor nodes) storing in storage positions a large amount of data which they have collected and consumers (e.g., base stations, users, and sensor nodes) then retrieving that data. The data storage strategy, including the decision as to where to place storage positions for producers and the provision of responses to consumers' queries, is critical in WSNs. A poorly designed data storage strategy will increase communication overheads, dissipate valuable energy and reduce the lifespan of battery-powered sensor networks. In contrast, a good storage strategy can tremendously reduce the energy consumption for data storage and retrieval, minimize query processing delays and prolong the lifespan of a sensor network. Further, more desirable are strategies that can place data adaptively so as to minimize the costs of storage and query.

Two main factors that impact data storage-related communication cost are the data rates of producers and consumers and their path distances to the storage node. The data rate of producers denotes the data producing rate from producers. The data rate of consumers denotes the data querying rate from consumers. The data rate usually does not change in a fixed application-specific time interval. For example, where there is only one producer and one consumer, data would be stored closer to the consumer rather than the producer when the query rate is higher than the data producing rate, and vise versa. In a real sensor network, the closer the storage node is to the producers and consumers, that is, the shorter the hop distance, the cheaper it is to store and query a fixed quantity of data. An effective way to do this is to place data adaptively according to network state (e.g., locations of nodes requesting data and their data rates) so that the communication cost is minimal once the data storage position is placed.

Although these two factors had been investigated, previous work mainly addressed the storage problem by treating the sensor network as a tree structure, in which the base station [10,15,19] is normally treated as the storage node or the only consumer. In the tree structure, the data rates of producers and the query rate of the base station are known or at least predictable and communication cost can be optimized as data storage placement is simply a response to data volumes. In a mesh network topology, however, there are potentially multiple producers and consumers all seeking to exploit one event simultaneously. In this scenario, some work has focused on the geographical locations [18] of producers and consumers but given no attention to the issue of the data rates [5,8,13]. Once the network topology is fixed, the storage node

* Corresponding author. Tel.: +852 27667270.
E-mail addresses: zcyu@fudan.edu.cn (Z. Yu), csbxiao@comp.polyu.edu.hk (B. Xiao), sgzhou@fudan.edu.cn (S. Zhou).

0140-3664/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2009.08.005

position is fixed too, which does not apply adaptive storage principle to reduce energy consumption in sensor networks. The drawback is that the storage node positions may not be location-aware nor adaptive to network state.

Rather than determining n storage locations[1] for an event, we focus on the single-node storage problem in a wireless sensor network with a mesh topology, where information collected from all producers is sent to a storage node and all consumers retrieve information from there. Similar to previous centralized data storage approaches like [10,13,19], the single location can be event-oriented, i.e., one storage node to store data related to a particular event. Data load is generally asymmetric in a mesh network topology, e.g., for the security purpose [20]. Such an unbalanced data volume causes an uneven energy consumption distribution among different sensor nodes. Since traffic load and energy consumption of each node are location-dependent and rate-dependent, the network lifetime can be shortened by nodes with greater energy consumption. Therefore, storage node placement scheme can have considerable impact on the network lifetime.

In this paper, we propose an optimal data storage (ODS) strategy in a wireless sensor network that allows the storage location to adaptively change, in response to both the geographical locations of producers and consumers and the data rates at which data are being exchanged. This strategy minimizes the energy consumption of the total network, and decreases the delay for message exchange between producers (or consumers) and the storage node. That is, storage position varies adaptively in response to data rates of nodes and their geographical locations. Specifically, we design and implement our adaptive data storage strategy to determine the total communication cost in the *one*-to-*one* (*one* producer and *one* consumer) and *many*-to-*many* (m producers and n consumers) models.

In the *one*-to-*one* model, ODS algorithm can generate an optimal storage location with minimum global communication cost. In the *many*-to-*many* model, multiple producer and consumer nodes can distribute in linear, grid, or mesh network topologies. Given each distribution, we propose distinct ODS algorithms to locate the globally optimum storage position. We show that to determine the optimal storage position, the ODS algorithm complexity is $O(1)$ in the *one*-to-*one* model, $O(m+n)$ in the linear and grid topologies for m producers and n consumers. In other words, to get an optimal data storage location for sensors regularly distributed, ODS only requires *constant* computation time when $m+n$ is much smaller than N, the total number of nodes in a sensor network. In the mesh network topology, ODS becomes a greedy algorithm and its time complexity is $O(N*(m+n))$. To reduce its computational overheads, we aim to get an approximation solution from the geometry theory and propose NDS, a near-optimal data storage strategy, an alternative to ODS. NDS reduces the computation complexity to be a fraction $\pi R^2/S$ of ODS where R is the sensor transmission range and S is the sensor distribution area.

We conduct extensive simulations to show that, compared to centralized data storage (CDS) [10,19] and geographic hash table (GHT) [13], ODS can greatly reduce the energy consumption as well as minimize delays in the data exchange process. NDS is also very efficient to generate the optimal storage position as ODS in more than 75% simulation cases.

The rest of this paper is organized as follows. Section 2 discusses related work. In Section 3, we describe terminology and assumptions to formulate the data storage problem to save the communication energy cost. In Section 4, we implement ODS in the *one*-to-*one* model and show its performance evaluation. Section 5 presents the ODS and NDS algorithms in the *many*-to-*many* model and Section 6 illustrates simulation results. Section 7 offers our conclusion.

[1] As denoted in [12], it is an NP-hard problem for the file multiple copies allocation in computer networks. Thus, in this paper, we limit our discussion for optimal but efficient storage location in a single node in wireless sensor networks.

2. Related work

Data storage in wireless sensor networks are addressed in either tree structure or mesh network topologies. Tree structures feature only one consumer (usually the base station at the root) and multiple producers and do not take geographical location information into account when determining data storage placement. In contrast, a mesh network involves multiple producers and consumers yet the dominant approach in the design of data storage to minimize communication overhead has been given to emphasize geographical locations with little attention to the data rates.

The tree structure offers a range of data storage strategies. Data can be stored and processed in a centralized server or in a distributed manner. In centralized data storage (CDS) approaches such as COUGAR [19] and TinyDB [10], all sensor nodes feed their data to the base station at the root of the tree. Scoop [4] does collect statistics about data, queries, and network conditions and uses them to dynamically change an in-network storage policy. However, it collects these statistics only periodically and uses a greedy algorithm to compute the optimal storage position. Such a try-and-test greedy algorithm is tremendously complex, making it infeasible for a large-scale sensor network. Sheng et al. [15] utilized data rates, query rates, and compression ratio to determine storage placement and introduced storage nodes to alleviate the heavy load of transmitting all data to a central place (the base station). They proposed the optimal placement of multiple storage nodes but it can be only applied in a tree topology. The local data storage (LDS) in the tree structure requires that sensor nodes store data locally in their own memory. Queries are flooded to all nodes in the network (or at least to all nodes that could possibly hold the relevant data) and nodes holding the appropriate data then reply. Apart from flooding, queries can also be disseminated by direct diffusion [7], in which query costs are reduced by using in-network aggregation.

In the mesh network topology, the typically proposed brokerage rules are data centric storage (DCS) [16] and geographic hash table (GHT) [13]. Both methods combine the idea of a distributed hash table (DHT) in peer-to-peer system with that of geographic naming and routing. They use geographical locations as reservoirs where data are hashed to and retrieved from. DIM [8] presents a distributed index for multi-dimensional data that uses special locality-preserving hash mapping. The hash mapping hashes nearby sensor data to the same node, and uses *k-d* tree to support range queries. DIFS [5] relies on geographic hash and *quad* tree structure for efficient index construction and range searches. DIMENSIONS [3] uses data compression and data aging to reduce redundancy caused by spatial and temporal correlation. The indexing approaches could be expensive for data storage because data can be sent far across the network and index itself can be difficult and complex to maintain.

Special path routing approaches provide a kind of information brokerage in which the producers store data not at a single node or its nearby neighbors but at nodes that follow a one-dimensional curve. The consumers travel along a set of nodes that follow another one-dimensional curve. When the two curves intersect, a consumer acquires the appropriate data. Each of these curves is drawn from a function of, respectively, the locations of producers and consumers, and not related to the type of data that is stored or queried. Examples include double ruling [14], landmark [2], and combs-needles-haystacks [9]. The common feature is that they take geographical locations into consideration but without data rates.

3. Problem formulation

In this section, we formulate the adaptive data storage problem by explaining the data storage process and involved communication overheads. We give terminology and assumptions to model the cost of data storage. Then, we define the optimal data storage strategy as to find the storage node position with the minimum cost.

3.1. Terminology and assumptions

Data storage strategies determine communication overheads because they specify how data are stored as well as how queries are routed to acquire the stored data. Nodes in a sensor network can exchange data, a process containing three phases:

1. *Producers* collect sensed data and, according to a predefined strategy, store them on a storage node in network.
2. *Consumers* send query requests to a relevant node. These query requests travel along routes defined by a routing rule.
3. *Storage* node which holds the appropriate data preprocesses the raw data and feedbacks it to consumers.

Fig. 1 illustrates such a scenario in which producers detect events and store critical data into a storage node. Consumers then issue queries to the storage node for acquiring relevant information.

We refer to the communication overheads in each of these three phases as, respectively, storage cost or $C_{Storage}$, diffusion cost or $C_{Diffusion}$, and reply cost or C_{Reply}. For simplicity of presentation, we sum the latter two and refer to them as query cost or C_{Query}, i.e., $C_{Query} = C_{Diffusion} + C_{Reply}$. The communication overheads are not only proportional to the number and size of data messages, as determined by the data rates, but also related to the transmission routes as determined by the locations of producers and consumers. In formulating the data storage problem, we have the following assumptions:

- All nodes except the base station are equivalent. The radio transmission range is R for all nodes. There are N nodes in the network and they are labeled $1, 2, \ldots, i, \ldots, N$. Let i be a node in the network, and its location is given as (x_i, y_i).

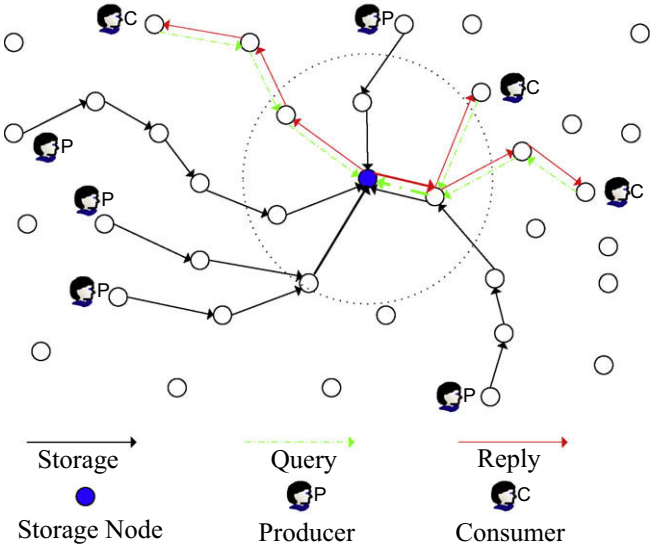

Fig. 1. An example of data storage in wireless sensor networks.

- The base station has the topology information of the network by monitoring and collecting the network statistics and can proceed with complicated calculation. It can compute the optimal storage position as requested in the *many*-to-*many* model.
- A communication edge e_{ij} exists between any pair of nodes i and j that are within radio range. To transmit s data units along the edge, the energy cost of the sender and receiver is $\sigma_{tr} + \delta_{tr}s$ and $\sigma_{re} + \delta_{re}s$, respectively [17], where σ_{tr} and σ_{re} are startup energy costs for transmitting and receiving a data packet, respectively, δ_{tr} and δ_{re} are energy cost for transmitting and receiving a data unit (byte), respectively. Compared with the energy cost for data transmission, the computation energy cost can be ignored. In this paper, we only take the energy cost in data transmission into consideration.
- An event occurs randomly at any place at any time. A node i sends event data to or collects data from a storage node k at the fixed rate $R(i)$ in a unit time interval.
- In a general case, there are m producers and n consumers linked to an event in the network simultaneously ($1 \leqslant m, n \leqslant N$ and $m + n < N$), and the storage node is node k. Let m producers be $p_1, p_2, \ldots, p_m$ and n consumers be $c_1, c_2, \ldots, c_n$. $C_{Storage}(i,k)$ denotes the storage cost when producer i sends event data to node k, and $C_{Query}(j,k)$ denotes the query cost when consumer j diffuses a query request to node k and acquires the relevant data. We study all the cost for data exchange in a unit time interval.

3.2. Energy cost formulation

We aim to determine the most energy-efficient storage position, that is, the position where the energy consumption associated with data storage, query diffusion and result reply is minimal. We develop a mathematical model to quantify, from all possible storage positions, the cost of storage and query and point out the minimal energy cost $min\{C_{Storage} + C_{Query}\}$ in the optimal storage strategy.

The storage cost $C_{Storage}(k)$ in a unit time interval is the sum of all costs of sending data to node k for storage from all producers. Whereas, the query cost $C_{Query}(k)$ is the sum of all costs of querying and acquiring the relevant data from all consumers.

$$C_{Storage}(k) = \sum_{i=p_1}^{p_m} C_{Storage}(i,k) \tag{1}$$

$$C_{Query}(k) = \sum_{j=c_1}^{c_n} C_{Query}(j,k) \tag{2}$$

The storage and query costs are only relevant to the data rate and hop distance in the data transmission path. The data rate is critical to energy consumption of data storage and access, the smaller the data rate is, the less it is to store and query a fixed quantity of data. To the storage node k, we have

$$C_{Storage}(i,k) = R(i) * h(i,k) \tag{3}$$

$$C_{Query}(j,k) = R(j) * h(j,k) \tag{4}$$

where $h(i,k)$ and $h(j,k)$ are the hop counts between the pair nodes (i,k) and (j,k), respectively.

The total cost $C_{Total}(k)$ is the sum of $C_{Storage}(k)$ and $C_{Query}(k)$ which is associated with producers storing the data at node k and consumers retrieving the appropriate data from there:

$$C_{Total}(k) = \sum_{i=p_1}^{p_m} R(i)^* h(i,k) + \sum_{j=c_1}^{c_n} R(j) * h(j,k) \tag{5}$$

Finding the optimal storage node, i.e., to get the minimal $C_{Total}(k)$ in Eq. (5), is a challenging problem because many factors can impact the storage position selection, e.g., data rates, multi-

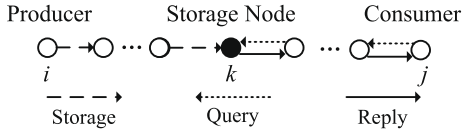

Fig. 2. The *one*-to-*one* model.

ple paths between two arbitrary nodes in the network, and the network topology. Although there may be multiple paths between two nodes, data are transferred along a fixed path for a certain application-specific time duration if the network does not change dramatically because the path between any two nodes is determined by the application's routing protocol.

The optimal storage position can be determined in the *one*-to-*one* model, or *many*-to-*many* model in a sensor network depending on the number of producers and consumers. In the *one*-to-*one* model, there is only one producer and one consumer. However, the *many*-to-*many* model allows the existence of multiple producers and multiple consumers. The optimal data storage position can be analyzed in linear, gird, and mesh network topology. Notice that the communication overhead is proportional to the message size and the distance between producer and consumer, and the distance refers not to the real distance but to hop counts in wireless sensor networks.

4. Optimal data storage and performance evaluation in one-to-one model

4.1. ODS algorithm

In this section, we propose the optimal data storage (ODS) algorithm in the *one*-to-*one* model (Fig. 2) to determine the optimal storage node k. There is only one producer i and one consumer j to broker information in the network. Because the network is connected, there must exist a shortest path P that connects producer and consumer through the storage node k. The path length is measured by the hop count between node i and j.

We assume the length of P is L and the distance between producer i and node k is a variable x. Therefore, the distance between consumer j and node k is $L - x$. According to Eq. (5), the total energy consumption choosing k as the storage node is

$$\begin{aligned} C_{Total}(k) &= R(i) * h(i,k) + R(j) * h(j,k) = R(i) * x + R(j) * (L - x) \\ &= R(j) * L + (R(i) - R(j)) * x \end{aligned} \quad (6)$$

From Eq. (6), an optimal storage location must minimize $C_{Total}(k)$ and thus we have Theorem 4.1.

Theorem 4.1. *Given a wireless sensor network with only one producer i and one consumer j, the optimal data storage position is at the consumer j when $R(i) < R(j)$, at the producer i when $R(i) > R(j)$, and at any node along the shortest path between the producer i and consumer j (inclusive) when $R(i) = R(j)$.*

From Theorem 4.1, we show Algorithm 1, ODS, given in pseudo-code below, which outputs the optimal data storage node in the *one*-to-*one* model. First, ODS compares the data rate via the input parameters. Then it directly outputs the optimal storage node according to Theorem 4.1 (*Lines 1–7*). Because ODS does not need to try and test each node in the network to output the optimal storage position, the algorithm complexity is $O(1)$.

Algorithm 1 ODS: The optimal data storage in *one*-to-*one* model

Input: Producer: node i and its data rate $R(i)$;
Consumer: node j and its data rate $R(j)$.
Output: The optimal storage node on the shortest path between i and j.
1: **if** $R(i) > R(j)$ **then**
2: Return the producer i;
3: **else if** $R(i) < R(j)$ **then**
4: Return the consumer j;
5: **else**
6: Return any node along the shortest path between i and j;
7: **end if**

4.2. Performance evaluation

We compare the performance of ODS with three alternative data storage strategies, Oracle, CDS, and GHT. Table 1 lists their underlying strategies in the *one*-to-*one* model. Oracle uses a try-and-test strategy that can determine the optimal storage node, that is, the node having the lowest communication energy cost according to Eq. (5). CDS [10,19] is a centralized strategy. It selects the center node in the network area as the base station which producers deliver data to and consumers query data from. GHT is an implementation of [13], in which the MD5 hash function is applied as a geographic hash function where the locations and data rates of producers and consumers are used as the input of the function. A 128-bit message digest is the output. We convert the digest into a long integer and get the data storage position in response to the length and width of the network area.

Our simulator set the radio range of a node to 50 m and the network area is limited to an area of 400 m × 400 m where 400 nodes are deployed randomly. The base station is placed at its center. We use a generic MAC-layer scheduling protocol that schedules transmissions to avoid collisions. In simulations, one node is randomly selected as the producer i and the other as the consumer j. Table 2 lists the simulation parameters described in Section 3.1.

Fig. 3 shows the energy consumption at different data rate of $R(i)$ when $R(j) = 60$ Bytes/s. We observe that ODS and Oracle have equivalent energy consumption, indicating that in the *one*-to-*one* model ODS is able to efficiently determine the optimal data storage node. Specifically, both of them have constant energy consumption when $R(i)$ is increased to 60 Bytes/s. This is because we select the consumer as the storage node when $R(i) \leqslant R(j)$ whereas the producer is the storage node when $R(i) > R(j)$ in ODS. From Fig. 3,

Table 1
The strategies tested in our simulation.

Strategy	Description
CDS	Send data to and get data from the base station
GHT	Use hash mapping (MD5 hash function)
ODS	Optimal data storage
Oracle	Try and test all nodes (benchmark)

Table 2
The simulation parameters.

Parameter	Value
Network area length	400 m
Network area width	400 m
Radio transmission range (R)	50 m
Data package size (s)	40 byte
Transmitting cost per-message (σ_{tr})	0.645 mJ
Transmitting cost per-byte (δ_{tr})	0.0144 mJ/byte
Receiving cost per-message (σ_{re})	0.387 mJ
Receiving cost per-byte (δ_{re})	0.00864 mJ/byte

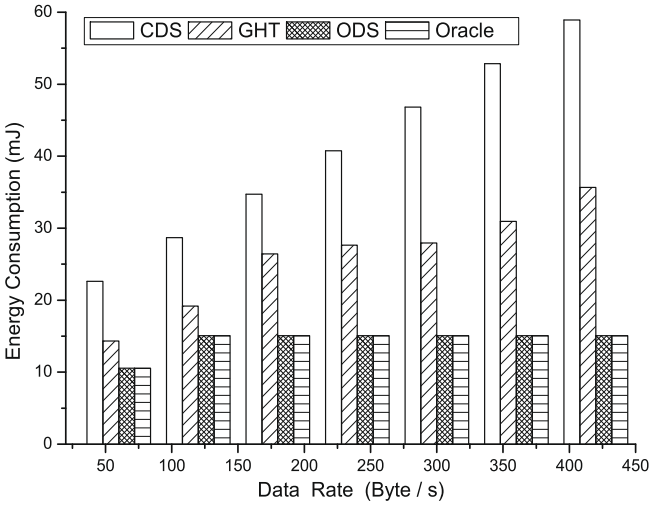

Fig. 3. Energy consumption when increasing the date rate of $R(i)$.

we can see that the higher the data rate is, the more energy consumption of GHT and CDS is, while ODS and Oracle can achieve the minimum, constant energy consumption.

Fig. 4 shows the energy consumption of different network size when $R(i) = R(j) = 60$ Bytes/s. Because we randomly select the producer and consumer, the energy consumption varies in each simulation. Both ODS and Oracle can obtain the same minimum energy consumption regardless of the number of nodes and the savings over GHT and CDS are significant. Notice that GHT is superior to CDS on average, this is because the storage node in CDS is fixed at the base station whereas in GHT it changes dynamically according to the hash function. We also tested the average delay of different strategies, represented by the total number of routing hops from producer i to consumer j. Fig. 5 shows the average delay when the number of nodes increased from 200 to 1000. The results display that ODS and Oracle have equal effect to minimize the number of hops of message transmission.

Fig. 6 shows the run time comparison between ODS and Oracle in the same environment. Given the locations of producer and consumer, Oracle verifies every node in a sensor network as the storage position and then select the one with the minimum energy cost. However, ODS has the run time complexity to be $O(1)$. No matter how many nodes distributed in a sensor network, the calculation time of the optimal storage node in ODS remains to be constant at a very low cost.

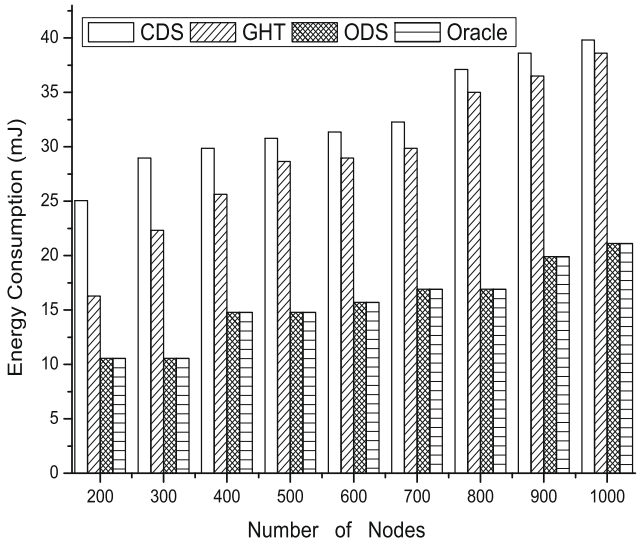

Fig. 4. Energy consumption in increased network size.

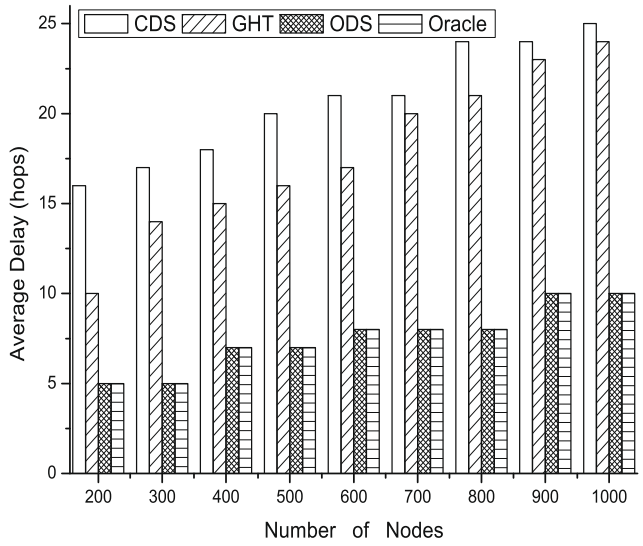

Fig. 5. Average delay in increased network size.

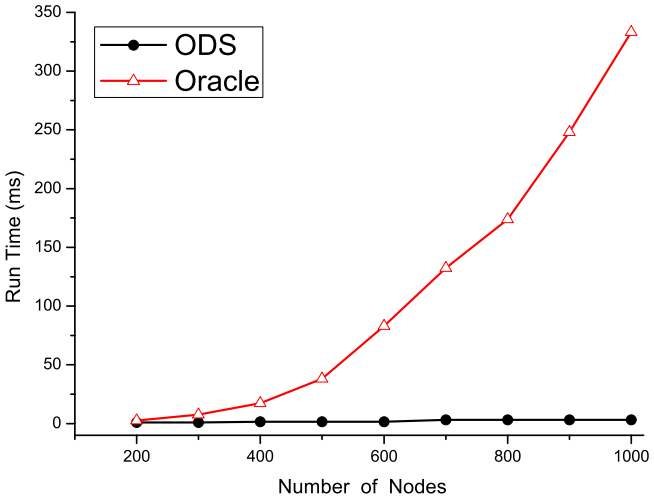

Fig. 6. Run time to determine the optimal storage location.

5. Optimal data storage in many-to-many model

In this section, we analyze the cost of data storage and query processing in a *many*-to-*many* (*many* producers to *many* consumers) model for linear, grid and mesh network topologies. Given m producers and n consumers, we first propose ODS algorithms to be applied in the linear, grid and mesh network topologies. To efficiently obtain a storage location, we then propose a near-optimal data storage (NDS) algorithm in the mesh network topology. Eq. (5) implies that, for a producer or a consumer, its contribution to the total cost only relates to its data rate and hop count to the storage node. Thus, in this section, we do not differentiate producers from consumers.

5.1. ODS in a linear network topology

We first analyze the total cost of implementing optimal data storage in a linear sensor network that consists of a set of sensor nodes placed along a long and narrow area. Each producer collects

the sampled data within its sensing range and relays data towards a storage node. Each consumer acquires the appropriate data from that storage node. For simplicity, each node relays data for nodes further away, i.e., node i also relays the data collected by nodes 1 to $i-1$ to the storage node, and does not proceed with data aggregation. Typical applications include traffic monitoring and border control.

Because all sensor nodes locate in a line, we use their coordinates to calculate the hop distance. Let x_i, x_j, and x_k be the coordinate of producer i, consumer j, and storage node k, respectively. Thus, $h(i,k)$ and $h(j,k)$ can be calculated as follows:

$$h(i,k) = |x_i - x_k|, \quad h(j,k) = |x_j - x_k| \tag{7}$$

Now we give the optimal storage position in a linear network with multiple producers and multiple consumers. We scan the linear network from two ends toward the midst simultaneously. A node with 0 data rate does not contribute any storage cost. Thus, when we reach a node whose data rate is zero, we simply skip it and move ahead. Since producers and consumers are treated equally and only their data processing rates count for the total storage cost, we should select the final storage position in the middle of the sensor distribution line to get balanced data rates at both sides. Algorithm 2 shows the pseudo-code of the optimal data storage strategy in a linear network topology. To reach a balanced position in the middle, the data rates at the left will be accumulated as $\sum Left$ and the right as $\sum Right$. $\sum Left$ will be increased when it is smaller than or equal to $\sum Right$, and vice versa. $\sum Left$ ($\sum Right$) gets an increased value at a node whose data rate is not zero. Whenever $\sum Left$ and $\sum Right$ arrive at positions with no data rates in-between, they are standing at node "i" and "j" for us to apply Theorem 4.1 to output the optimal data storage node. Because we scan the linear network once and only count producers and consumers, the complexity of this algorithm is $O(m+n)$ where m is the number of producers and n for consumers.

Algorithm 2 ODS-Linear: data storage in linear topology

Input: N: The total number of nodes in a linear topology;
m: Producers $p_1,\ldots,p_m$, their positions are $L(p_1),\ldots,L(p_m)$ and their data producing rates are $R(p_1),\ldots,R(p_m)$;
n: Consumers $c_1,\ldots,c_n$, their positions are $L(c_1),\ldots,L(c_n)$ and their data querying rates are $R(c_1),\ldots,R(c_n)$.
Output: The optimal storage node.

```
1: i = 1; j = N; ∑Left = ∑Right = 0;
2: while i < j do
3:   if all nodes between i and j are with data rate 0 then
4:     Break;
5:   end if
6:   if ∑Left ⩽ ∑Right then
7:     if R(i) = 0 then
8:       i = i + 1; Continue;
9:     else
10:      ∑Left += R(i);
11:      i = i + 1;
12:    end if
13:  else {∑Left > ∑Right}
14:    if R(j) = 0 then
15:      j = j − 1; Continue;
16:    else
17:      ∑Right += R(j);
18:      j = j − 1;
19:    end if
20:  end if
21: end while
22: Apply Algorithm 1 to determine the storage node where i and j
    are positions of producer and consumer, and ∑Left, ∑Right are
    their data rates, respectively.
```

Theorem 5.1. *Algorithm 2 outputs the optimal data storage position in a linear network topology.*

Proof. In Algorithm 2, *while* loop will be broken when Fig. 7 is met. i and j are two nodes with non-zero data rates. Between nodes i and j, all nodes are with the data rate 0. As the input to Algorithm 1, i can represent a producer with the sum data rate $\sum Left$ and j can represent a consumer with the sum data rate $\sum Right$. According to Algorithm 1, when $\sum Left > \sum Right$, the optimal storage node is at node i; when $\sum Left < \sum Right$, the optimal storage node is at node j; when $\sum Left = \sum Right$, any node in-between nodes i and j can be the optimal storage node. Here we only prove the case when $\sum Left > \sum Right$ while other cases can be proved similarly.

When the storage position is at node i, assume the total cost is $C_{Total}(i)$. Given that $\sum Left > \sum Right$, according to Algorithm 2, we have $\sum Left - R(i) \leqslant \sum Right$. Given a storage node to the left of i with a hop count d ($d > 0$), the new total cost becomes

$$\begin{aligned} C_{Total}(left_d) &\geqslant C_{Total}(i) - \left(\sum Left - R(i)\right) * d + (R(i) + \sum Right) * d \\ &= C_{Total}(i) + \left(\sum Right + R(i) - \sum Left\right) * d + R(i) * d \geqslant C_{Total}(i) \\ &\quad + R(i) * d > C_{Total}(i) \end{aligned} \tag{8}$$

Eq. (8) can be true when all nodes are with 0 data rate who are within d hop counts to the left of node i. Similarly, we can show that the storage node to the right of node i with a hop count d ($d > 0$) will also result in the total cost increase. □

5.2. ODS in a grid network topology

In this section, we propose the ODS for sensor nodes distributed in a grid network topology. Fig. 8 shows a lattice area with $\sqrt{N} * \sqrt{N}$ regular grid comprising N nodes. Each node has four neighbors adjacent to it. Such sensor network topology can be obtained by nodes manually distributed.

The position of a node i in the grid has the coordinates (x_i, y_i). Assume the coordinates of storage node k is (x_k, y_k) in the grid topology, Thus, $h(i,k)$ can be calculated as follows:

$$h(i,k) = |x_i - x_k| + |y_i - y_k| \tag{9}$$

ODS should select a storage position adaptive to the positions of the producers and consumers and their data rates. Once the storage position is determined, we can easily calculate the hop distance according to Eq. (9). A greedy algorithm that tests every grid position can generate the optimal position by comparisons. However, it is inefficient with complexity $O(N*(m+n))$. The grid topology has sensor nodes regularly distributed in a two-dimensional area. Thus, one approach is to decompose two-dimensional gird topology into two one-dimensional linear topologies. The optimal storage positions should exist in nodes whose storage coordinates are optimal in each dimension.

Algorithm 3 shows the ODS strategy in the grid network topology. To get the optimal storage position in each dimension, we project the data rates in the grid to x-axis and y-axis. For a node i, we project its data rate $R(i)$ to its x-coordinate and y-coordinate on x-axis and y-axis, respectively. The data rate can be accumulated on a

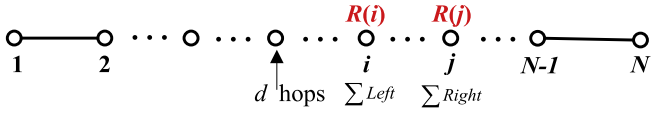

Fig. 7. Assume the optimal storage node at d hops to the left of $\sum Left$.

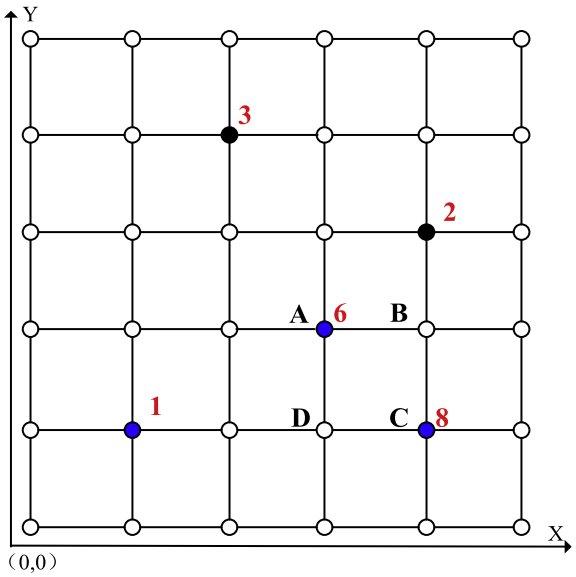

Fig. 8. The grid sensor network model.

coordinate, i.e., $R_x(x_i)+=R(i)$, $R_y(y_i)+=R(i)$. For example, there are two nodes whose x-coordinate is 4 in Fig 8. Thus, the data rate of the position where x-coordinate is 4 is 10 = 2 + 8 on the x-axis. After projecting data rates of producers and consumers to x-axis and y-axis, we can apply Algorithm 2 to obtain optimal storage positions in each dimension. In each dimension, there could be many such positions. Suppose we store them in arrays X and Y, respectively. Algorithm 3 will return a node whose first coordinate from X and second coordinate from Y. Theorem 5.2 shows the correctness of Algorithm 3 to return the optimal data storage position. In each dimension, there will be at most $m+n$ coordinates with data rates if there are m producers and n consumers in the network. Hence, the computation complexity in x-axis and y-axis is $O(m+n)$. The complexity of Algorithm 3 is $2\cdot O(m+n)$ for the computation at *Lines 11 and 12*.

Algorithm 3 ODS-Grid: data storage in grid topology

Input: N: The total number of nodes in a grid topology;
m: Producers $p_1,\ldots,p_m$, their positions are $G(x_{p_1},y_{p_1}),\ldots,G(x_{p_m},y_{p_m})$ and their data producing rates are $R(p_1),\ldots,R(p_m)$, respectively;
n: Consumers $c_1,\ldots,c_n$, their positions are $G(x_{c_1},y_{c_1}),\ldots,G(x_{c_n},y_{c_n})$ and their data querying rates are $R(c_1),\ldots,R(c_n)$, respectively.
Output: The optimal storage node.
1: Reset $R_x(i)=0, i=0,\ldots,\sqrt{N}$ where i represents a node in the x-axis;
2: Reset $R_y(j)=0. j=0,\ldots,\sqrt{N}$ where j represents a node in the y-axis;
3: **for** $i=p_1$ to p_m **do**
4: $R_x(x_i)+=R(i)$;
5: $R_y(y_i)+=R(i)$;
6: **end for**
7: **for** $j=c_1$ to c_n
8: $R_x(x_j)+=R(j)$;
9: $R_y(y_j)+=R(j)$;
10: **end for**
11: Apply Algorithm 2 to obtain optimal storage positions on the projected x-axis and store them in an array X;
12: Apply Algorithm 2 to obtain optimal storage positions on the projected y-axis and store them in an array Y;
13: Return a node whose x-coordinate in X and y-coordinate in Y.

Theorem 5.2. *Algorithm 3 outputs the optimal data storage positions in a grid network topology.*

Proof. Because the two dimensional grid topology can be decomposed into x-axis and y-axis, the optimal storage position must achieve the minimal cost in each axis. Algorithm 3 can obtain such optimal positions in both axes by applying Algorithm 2 twice, which ensures the optimal data storage positions in each dimension. □

We give an example to illustrate Algorithm 3. The data rates in Fig. 8 can be projected to x-axis and y-axis, respectively, as shown in Fig. 9. We scan the projected x-axis and get the optimal positions $x=3$ and $x=4$. Similarly, we get the optimal storage positions $y=2$ on the y-axis. Thus, the possible optimal positions will be $A(x=3,y=2)$ and $B(x=4,y=2)$. In order to verify its correctness, we calculate the cost of different nodes as follows: $C_{Total}(A)=3*3+2*2+8*2+1*3=32$, $C_{Total}(B)=32$, $C_{Total}(C)=34$, $C_{Total}(D)=34$. The results show that nodes A and B are both optimal storage nodes.

5.3. ODS in a mesh network topology

In this section, we present ODS in a mesh network topology. Sensor nodes can form a mesh topology where nodes are connected in a random distribution in an area. We abstract the nodes as the vertex set V and the links as edge set E. Thus the wireless sensor network constructs a connected graph $G=(V,E)$ in which each edge (u,v) in E has a cost $c(u,v)$, which denotes the cost of storage or query. Based on Theorem 4.1, the most energy efficient path between pairwise nodes is the shortest path. If we further regard the product of the packet size and data rate as the cost of the edge, the problem becomes how to determine a storage node k such that the sum of the costs of the shortest path from the storage node to every producer and consumer is minimal according to Eq. (5). In the shortest path between a pair of nodes (e.g., i and k), the cost is the same to every edge and can be normalized as the data rate, assuming the packet size is a unit. Thus, the cost of a path is the product of the data rate and hop counts in the path.

To efficiently determine the optimal storage position is a challenge problem for nodes distributed in a mesh topology. There are $m+n$ variables $h(p_1,k),\ldots,h(p_m,k)$, $h(c_1,k),\ldots,h(c_n,k)$ in Eq. (5). It is not obvious to resolve the minimal $C_{Total}(k)$ among all possible nodes with multiple variables. Given the random distribution of nodes, the Euclidean distance of a pair of nodes is not proportional to their hop counts, which makes the problem more complicated. For simplicity, we consider the problem in the plane geometry and assume the shortest path is proportional to the Euclidean distance. Then $h(i,k)$ and $h(j,k)$ can be written as follows:

$$h(i,k)=\sqrt{(x_i-x_k)^2+(y_i-y_k)^2}$$
$$h(j,k)=\sqrt{(x_j-x_k)^2+(y_j-y_k)^2} \quad (10)$$

According to Eq. (5), the problem of finding the optimal storage node converts to 1-median problem. It can be formulated as follows: given $m+n$ points $p_i(x_i,y_i)$, finding a point $p(x_k,y_k)$ such that the sum of the Euclidean distance from all points to point p is

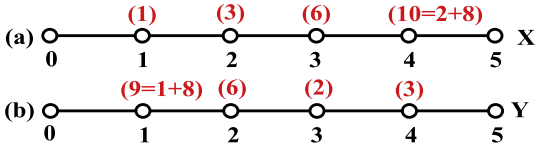

Fig. 9. The projection of x-, y-axis from the grid.

minimal. The 1-median problem is NP-hard [11]. There are many approaches to acquire approximation solutions. Arora et al. [1] proposed an $(1+1/c)$-Approximation schemes with computing complexity $O(n^{O(c+1)})$ where c is a constant specified by user. Har et al. [6] proposed an $(1+\varepsilon)$-Approximation algorithm via Coresets with computing complexity $O(\varepsilon^{-2}\lg n)$ where ε is parameter defined by user.

Different from the 1-median problem, the number of nodes (possible positions) in a sensor network is limited once the network is deployed in the optimal storage position problem. Thus, we can try and test every node and select the one as the storage position whose storage cost is minimal. In the following, we describe the optimal data storage algorithm (ODS-Mesh) in a mesh topology network, which is a greedy algorithm for determining the optimal storage node. ODS-Mesh, given in the pseudo-code below, selects from among all nodes the optimal node, which has the lowest storage cost according to Eq. (5). ODS-Mesh is a try-and-test approach, adopting a greedy policy by verifying every node in the network. Algorithm 4 finds the optimal position in the *for* loop over all possible nodes, computing the associated cost and selecting the node with the minimal cost as the storage node (*Lines 2–15*). Totally, there are two types of cost: $C_{Storage}$, which arises from producers sending data to the storage node and is calculated according to Eq. (1) (*Lines 4–6*), and C_{Query}, which arises from the consumers querying data from the storage node and is calculated according to Eq. (2) (*Lines 7–9*). $h(i,k)$ and $h(j,k)$ remain to be the hop counts between (i,k) and (j,k).

Algorithm 4 ODS-Mesh: data storage in mesh topology

Input:N: The total number of nodes in a mesh network;
m: Producers $p_1,\ldots,p_m$ and their data producing rates are $R(p_1),\ldots,R(p_m)$, respectively;
n: Consumers $c_1,\ldots,c_n$ and their data querying rates are $R(c_1),\ldots,R(c_n)$, respectively.
Output: The optimal storage node *ods*.
1: Initialize the minimal cost $C_{min}=+\infty$;
2: **for** $k=1$ to N **do**
3: Set $C_{Storage}(k), C_{Query}(k), C_{Total}(k)$ to zero;
4: **for** $i=p_1$ to p_m
5: $C_{Storage}(k)+=R(i)*h(i,k)$;
6: **end for**
7: **for** $j=c_1$ to c_n
8: $C_{Query}(k)+=R(j)*h(j,k)$;
9: **end for**
10: $C_{Total}(k)+=C_{Storage}(k)+C_{Query}(k)$;
11: **if** $C_{Total}(k)<C_{min}$ **then**
12: $C_{min}=C_{Total}(k)$
13: $ods=k$
14: **end if**
15: **end for**
16: Select *ods* as the storage node $C_{Total}(ods)=min_{1\leqslant k\leqslant N}\{C_{Total}(k)\}$;
17: Return the optimal storage node *ods*.

The complexity of ODS-Mesh is $O(N*(m+n))$ where N, m, and n are, respectively, the number of nodes, producers and consumers. N is the primary factor affecting the complexity in a given network topology. Such a try-and-test approach is inefficient when the network consists of a large number of sensor nodes. It does not take into account the locations of producers and consumers as well as data rates, but rather blindly verifies every node. This is not only time-consuming, but also impractical in large scale networks.

5.4. NDS in a mesh network topology

To reduce the computational complexity of ODS-Mesh, we give an approximation approach in the mesh network topology to find a NDS position. The new approach is built on the basis that when sensor nodes are uniformly and densely distributed in an area, the hop count between pairwise nodes can be well approximated by using their Euclidean distance. Although the hop count may be affected by many factors, like Euclidean distance, network density, routing strategy, topology regularity, etc., we still can use the Euclidean distance as a rough estimation. Normally, hop counts becomes larger along with the increment of Euclidean distance of pairwise nodes. For finding $min_{1\leqslant k\leqslant N}\{C_{Total}(k)\}$, we introduce the following function $f(x,y)$ as an approximation to the total storage cost, in which (x_{p_i},y_{p_i}) and (x_{c_j},y_{c_j}) denotes the location of producer p_i and consumer c_j, respectively, and (x,y) denotes the coordinate of the storage node k. $h(i,k)$ and $h(j,k)$ represent the Euclidean distance between pairwise nodes (i,k) and (j,k).

$$\begin{aligned} f(x,y) &= \sum_{i=p_1}^{p_m} R(i)*h^2(i,k)+\sum_{j=c_1}^{c_n} R(j)*h^2(j,k) \\ &= \sum_{i=p_1}^{p_m} R(i)*\left((x-x_i)^2+(y-y_i)^2\right)+\sum_{j=c_1}^{c_n} R(j) \\ &\quad *\left((x-x_j)^2+(y-y_j)^2\right) \end{aligned} \tag{11}$$

The problem of finding a near optimal storage position converts to determining the position (x_k,y_k) such that the value of the function $f(x_k,y_k)$ is minimal. Because $f(x,y)$ is a continuous function with two variables x and y confined to a bounded region in the xy-plane, we can resolve the local minimum value by

$$\begin{cases} f'_x(x,y)=0 \\ f'_y(x,y)=0 \end{cases} \tag{12}$$

We continue to calculate the second partial derivatives of the function, and get $f'_{xx}(x,y)=f'_{yy}(x,y)=2\left(\sum_{i=1}^{m}R(i)+\sum_{j=1}^{n}R(j)\right)>0$, $f'_{xy}(x,y)=0$. To obtain the extremum (x_k,y_k) of $f(x,y)$, we define $d=f'_{xx}(x_k,y_k)f'_{yy}(x_k,y_k)-\left[f'_{xy}(x_k,y_k)\right]^2$. If $d>0$ and $f'_{xx}(x_k,y_k)>0$, then (x_k,y_k) is a relative minimum. Therefore, we derive the following position (x_k,y_k) where Function (11) reaches its extremum. Such position (x_k,y_k) can be a reference point for the storage.

$$\begin{cases} x_k=\dfrac{\sum_{i=p_1}^{p_m}R(i)*x_i+\sum_{j=c_1}^{c_n}R(j)*x_j}{\sum_{i=p_1}^{p_m}R(i)+\sum_{j=c_1}^{c_n}R(j)} \\ y_k=\dfrac{\sum_{i=p_1}^{p_m}R(i)*y_i+\sum_{j=c_1}^{c_n}R(j)*y_j}{\sum_{i=p_1}^{p_m}R(i)+\sum_{j=c_1}^{c_n}R(j)} \end{cases} \tag{13}$$

Now we propose NDS, the near-optimal data storage algorithm, to find a storage position that can be or close to the optimal position. NDS avoids the heavy computation in a try-and-test strategy such as ODS-Mesh because it starts from a reference position heuristically and verifies only a small part of nodes. The key idea of NDS is to first find a reference position, which could be within one-hop distance to the real optimal storage position with a high probability. It does this by utilizing (x_k,y_k) in Eq. (13) as the reference position. Centered from it, a local optimal node within one-hop distance will serve as the storage node.

Algorithm 5 provides the pseudo-code for NDS, in which the reference position is calculated according to Eq. (13) (*Line 1*). Then the node that is nearest to the reference position is selected as starting node (*Line 2*). From this node the local optimal storage node is found among its one-hop neighbors (*Lines 5–18*). All nodes that are one-hop away from the starting node are tested and their costs are computed according to Eq. (5). Finally, the lowest cost node is selected as the storage position (*Line 19*). Notice that we initially select the starting node as the storage node. This avoids missing itself as the local optimal one (*Line 3*). In the algorithm,

$h(i,k)$ and $h(j,k)$ denote the hop counts between (i,k) and (j,k), respectively.

Algorithm 5 NDS: Near-optimal data storage in a mesh topology

Input: N: The total number of nodes in a mesh network;
m: Producers $p_1,\ldots,p_m$, their positions are $G(x_{p_1},y_{p_1}),\ldots,G(x_{p_m},y_{p_m})$ and their data producing rates are $R(p_1),\ldots,R(p_m)$, respectively;
n: Consumers $c_1,\ldots,c_n$, their positions are $G(x_{c_1},y_{c_1}),\ldots,G(x_{c_n},y_{c_n})$ and their data querying rates are $R(c_1),\ldots,R(c_n)$, respectively.
Output: The near-optimal storage node.
1: Calculate the reference position (x_0,y_0) according to Eq. (13);
2: Find the starting node N_{start} nearest to the reference position;
3: Set near-optimal storage node $nds = N_{start}$ and let C_{min} be $C_{Total}(N_{start})$;
4: Calculate the number of neighbors, $NUM_{neighbor}$, of the node N_{start};
5: **for** $k = 1$ to $NUM_{neighbor}$ **do**
6: Set $C_{Storage}(k), C_{Query}(k), C_{Total}(k)$ to zero;
7: **for** $i = p_1$ to p_m **do**
8: $C_{Storage}(k) += R(i) * h(i,k)$;
9: **end for**
10: **for** $j = c_1$ to c_n **do**
11: $C_{Query}(k) += R(j) * h(j,k)$;
12: **end for**
13: $C_{Total}(k) += C_{Storage}(k) + C_{Query}(k)$;
14: **if** $C_{Total}(k) < C_{min}$
15: $C_{min} = C_{Total}(k)$
16: $nds = k$
17: **end if**
18: **end for**
19: Return nds as the storage node.

An advantage of Algorithm 5 is that it is able to make use of reference position to greatly reduce the search space to a one-hop range. Meanwhile, how close the returned storage node nds to the optimal storage position depends on the approximation of the calculated reference position to the "median" of all producers and consumers. Simulation results in Section 6 demonstrate that the reference position (x_k,y_k) in Eq. (13) is a good approximation and there are more than 75% cases to get the optimal storage position by running NDS algorithm. To analyze its complexity, we assume that the area size of the network is S, the radio transmission range of a sensor node is R, and the total number of nodes is N. The average number of nodes that are one-hop away from the starting node is $N\pi R^2/S$. Since NDS only needs to try-and-test each node within neighbors in an area of πR^2, the complexity of NDS (Algorithm 5) is a fraction $\pi R^2/S$ of ODS-Mesh (Algorithm 4), which is $O(\pi R^2/S * N * (m+n))$.

6. Performance evaluation in many-to-many model

In this section, we evaluate the performance of ODS and NDS in the common situation where there are m producers and n consumers forming a mesh topology. Note that ODS denotes ODS-Mesh algorithm. We do not provide the simulation results of ODS-Linear and ODS-Grid algorithms because we have already proved their generated optimal data storage positions. After confirming the viability of our design, we compared our strategy with other alternative strategies CDS [10,19] and GHT [13], and also evaluated the effect of different parameters, such as the number of nodes (N), producers (m) and consumers (n), the average delay between the producers and consumers, algorithm run time to compute storage position. The simulation was conducted as follows: a node (e.g., the base station) collected all parameters of the network and computed a storage position, then the producers and consumers brokered information through the storage node. Because we are interested in evaluating long-term performance for different data storage strategies with continuous queries, each test was run for 100 simulation rounds with different initial topology. All experiments, unless otherwise denoted, used the same basic parameters.

In each testing, we randomly selected a number of producers and consumers among nodes and arbitrarily set their data rates with a range between 40 Bytes/s and 400 Bytes/s. Let d denote the average node degree that is the average number of neighbors in a one-hop range of the starting node. Let $AVG(d)$ denote the average node degree over 100 rounds. In our simulations, d has a value between 3 and 6. In every 100 simulation rounds, we recorded the times that NDS can produce the same optimal storage position as ODS, denoted as *EQU* (times). The *DIF* (percentage) displays the over-consumed energy of NDS on ODS, defined as $\frac{E_{NDS}-E_{ODS}}{E_{ODS}} * 100\%$.

Fig. 10 shows the average energy consumption for different numbers of consumers where $N = 400$ and $m = 150$. In order to evaluate the impact from consumers, we varied their number between 1 and 50 while fixing the number of producers at 150. The consumer number increment will increase the amount of energy cost for all storage strategies, yet it is smaller for ODS and NDS. We note in particular that the average energy consumption of NDS is very close to that of ODS as can be seen from the overlap of their curves. This is because when nodes are uniformly distributed in the network area, the one-hop coverage area of the starting node in most cases contains the optimal storage node. Table 3 lays out the details between ODS and NDS where the last two columns illustrate their average energy consumption in 100 testing rounds. Within 100 tests, there are over 75% cases that NDS can produce the optimal storage position as ODS does. This percentage is still true in following simulations, showing the accuracy of NDS to be a good approximation algorithm to ODS.

Fig. 11 shows the average energy consumption when the number of producers increases from 10 to 200 when $N = 400$ and $n = 10$. A larger number of producers will dramatically increase the average energy consumed by every strategy. Compared to CDS and GHT, ODS and NDS maintain lower energy consumption. NDS can almost achieve the same less energy cost as ODS do when producers send data to the selected storage node and consumers query data from it. Table 4 provides the data in detail. The dynamic storage node in

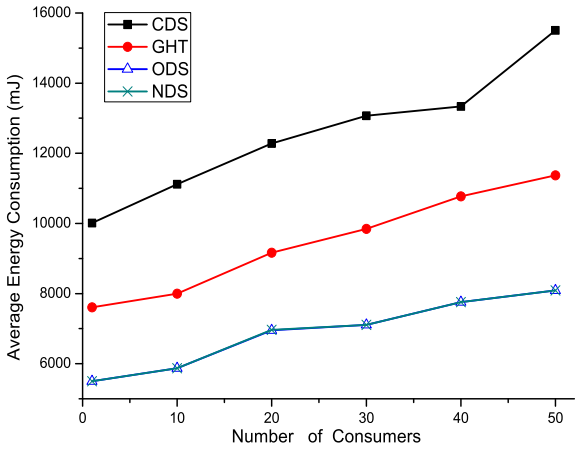

Fig. 10. Average energy consumption in increased number of consumers.

Table 3
The energy consumption ($N = 400$, $m = 150$).

n	$AVG(d)$	*EQU*	*DIF* (%)	E_{ODS} (mJ)	E_{NDS} (mJ)
1	3.79	85	0.090	5497.95	5502.89
10	3.72	99	0.004	5873.96	5874.18
20	3.32	75	0.206	6951.82	6966.11
30	3.88	89	0.021	7104.63	7106.14
40	3.84	87	0.015	7754.70	7755.89
50	5.62	86	0.002	8088.51	8088.69

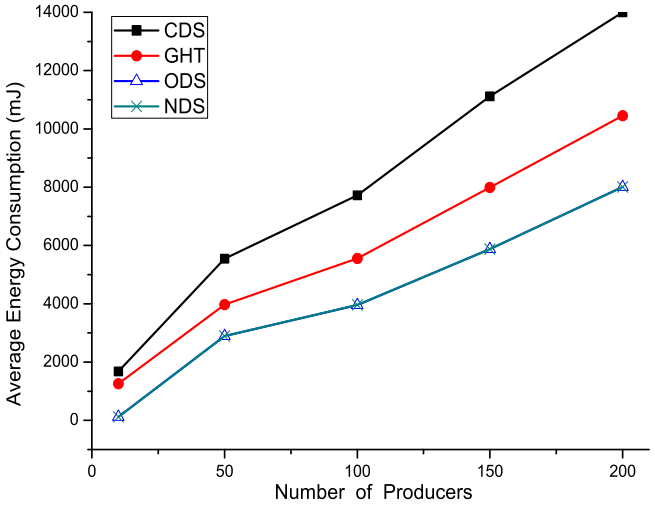

Fig. 11. Average energy consumption in increased number of producers.

Table 4
The energy consumption ($N = 400$, $n = 10$).

m	$AVG(d)$	EQU	DIF (%)	E_{ODS} (mJ)	E_{NDS} (mJ)
10	3.72	83	0.085	118.25	118.35
50	5.14	88	0.057	2891.46	2893.12
100	5.47	90	0.090	3958.09	3961.65
150	3.72	99	0.004	5873.96	5874.18
200	4.36	88	0.059	8007.49	8012.21

GHT makes it a better choice than CDS to reduce the data transmission cost, as shown in Figs. 10 and 11.

Fig. 12 shows the average energy consumption in increased network size when the number of producers and consumers are fixed at $m = 50$ and $n = 1$, respectively. We can see that the energy consumption does not necessarily increase with the number of nodes distributed in a sensor network. This is because the data rates and query rates are set randomly. However, ODS and NDS use much less energy than other approaches. From Fig. 12, we can witness the height of NDS in the histogram is almost equal to ODS. For showing the detail, we label the $AVG(d)$ and EQU for 100 rounds in the figure. Table 5 shows the relative value for ODS and NDS in detail. The energy cost difference between ODS and NDS is very trivial. Even the maximal DIF is less than 0.5%, which can be neglected in most cases.

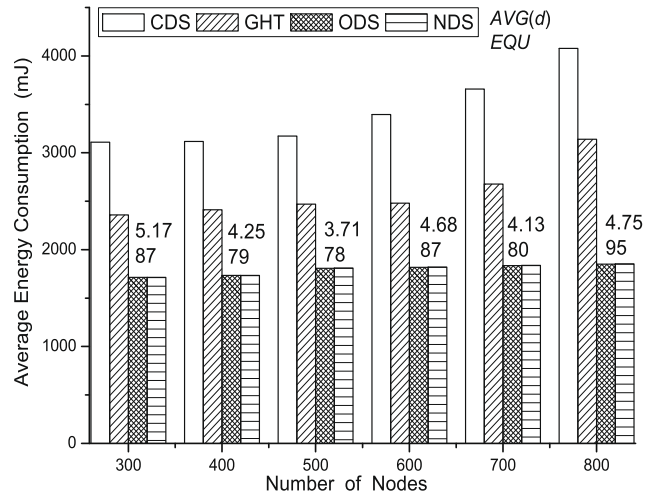

Fig. 12. Average energy consumption in increased network size.

Table 5
The energy consumption ($m = 50$, $n = 1$).

N	$AVG(d)$	EQU	DIF (%)	E_{ODS} (mJ)	E_{NDS} (mJ)
300	5.17	87	0.056	1713.49	1714.45
400	4.25	79	0.006	1735.01	1735.13
500	3.71	78	0.210	1806.95	1810.73
600	4.68	87	0.151	1816.76	1819.49
700	4.13	80	0.231	1834.93	1839.16
800	4.75	95	0.179	1849.68	1852.99

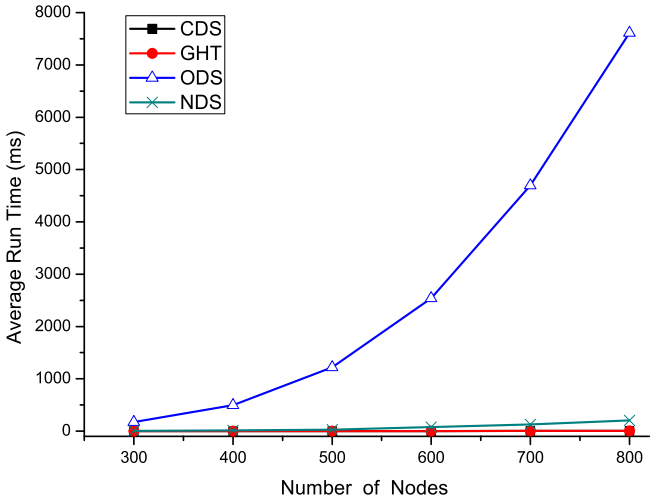

Fig. 13. Average run time in increased network size.

In the following, we will illustrate the computing complexity of different algorithms by means of comparing the run time in the same testing environment. We tested ODS, NDS, CDS, and GHT in the same PC with *Pentium IV 1.70 GHz* CPU and *256 MB* main memory. Fig. 13 shows the average run time to get the storage position with different number of nodes when $m = 200$, $n = 50$ in 100 simulation rounds. CDS and GHT can get the storage position directly. They have almost the same run time and their curves overlap. Since ODS needs to try-and-test each sensor node, its run time increases tremendously with the number of nodes in the network, which makes it infeasible in a very large-scale network. In contrast, NDS increases much mildly because it only needs to test neighboring nodes.

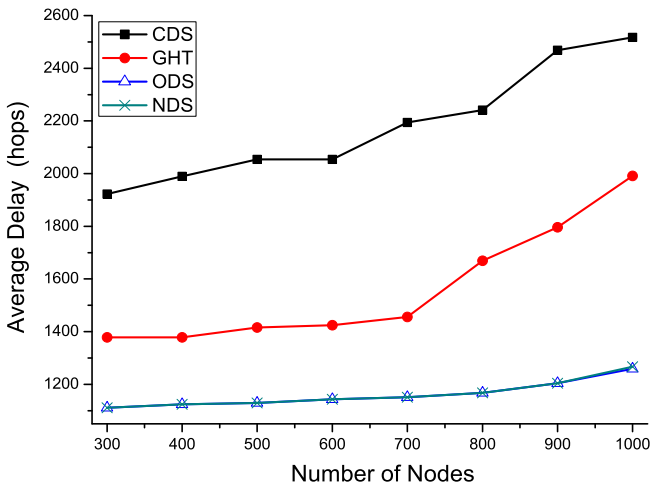

Fig. 14. Average delay in increased network size.

The delay between producers and consumers also plays an important role in the data storage process in wireless sensor networks. We measure the average delay using the total number of routing hops from the producers and consumers to the storage node in a deployed network. This is a parameter which can be used to reflect the total traffic to deliver an event to multiple consumers. Fig. 14 shows the average delay for nodes between 400 and 1000 and $m = 200$, $n = 50$ in 100 simulation rounds. The curves of ODS and NDS are very close and the average delay of the producers and consumers in both ODS and NDS is approximately 4.4 hops ($1100/(200 + 50) \approx 4.4$), which is much less than the delay of CDS and GHT.

7. Conclusions

This paper integrated data rates into the data storage problem to reduce the total energy cost in the wireless sensor network. Multiple producers and consumers may exist in the network to make the problem more complicated. To efficiently solve the storage problem, we proposed two novel data storage strategies, optimal data storage (ODS) and near-optimal data storage (NDS). They can both minimize energy consumption by choosing the storage node adaptively by taking data rates of producers, query rates of consumers, and their geographic locations into consideration. Theoretical analysis and simulation results show that ODS is able to compute the global optimal storage position while NDS can find a local optimal one.

Acknowledgements

This work was partially supported by HK RGC PolyU 5322/08E, China NSFC under Grant Nos. 60803161 and 60873070, and 863 Program under Grant No. 2009AA01Z135.

References

[1] S. Arora, P. Raghavan, S. Rao, Polynomial time approximation schemes for euclidean *k*-medians and related problems, in: Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98), Dallas, TX, USA, May 1998, pp. 106–113.

[2] Q. Fang, J. Gao, L.J. Guibas, Landmark-based information storage and retrieval in sensor networks, in: Proceedings of the 25th Conference of the IEEE Communication Society (INFOCOM'06), Barcelona, Catalunya, Spain, April 2006, pp. 1–12.

[3] D. Ganesan, D. Estrin, J. Heidemann, Dimensions: why do we need a new data handling architecture for sensor networks?, Computer Communication Review 33 (1) (2003) 143–148

[4] T.M. Gil, S. Madden, Scoop: an adaptive indexing scheme for stored data in sensor networks, in: Proceedings of the 23rd International Conference on Data Engineering (ICDE'07), 2007, pp. 1345–1349.

[5] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, S. Shenker. Difs: a distributed index for features in sensor networks, in: Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03), Anchorage, Alaska, USA, May 2003, pp. 163–173.

[6] S. Har-Peled, S. Mazumdar, Coresets for *k*-means and *k*-median clustering and their applications, in: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04), Chicago, IL, USA, June 2004, pp. 291–300.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom'00), Boston, Massachusetts, USA, August 2000, pp. 56–67.

[8] X. Li, Y. Kim, R. Govindan, W. Hong, Multi-dimensional range queries in sensor networks, in: Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (Sensys'03), Los Angeles, California, USA, November 2003, pp. 63–75.

[9] X. Liu, Q. Huang, Y. Zhang, Combs, needles, haystacks: balancing push and pull for discovery in large scale sensor networks, in: Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04), Baltimore, MD, USA, November 2004, pp. 122–133.

[10] S. Madden, M. Franklin, J. Hellerstein, W. Hong, Tinydb: an acquisitional query processing system for sensor networks, ACM Transactions on Database Systems (TODS) 30 (1) (2005) 122–173.

[11] N. Megiddo, A. Tamir, On the complexity of locating linear facilities in the plane, Operations Research Letters 5 (1) (1982) 194–197.

[12] K. Murthy, J. Kam, M.S. Krishnamoorthy, An approximation algorithm to the file allocation problem in computer networks, in: Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, 1983, pp. 258–266.

[13] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, S. Shenker. Ght: a geographic hash table for data-centric storage, in: Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta, Georgia, USA, September 2002, pp. 56–67.

[14] R. Sarkar, X. Zhu, J. Gao, Double rulings for information brokerage in sensor networks, in: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom'06), Los Angeles, California, USA, September 2006, pp. 286–297.

[15] B. Sheng, Q. Li, W. Mao, Data storage placement in sensor networks, in: Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'06), 2006, pp. 344–355.

[16] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, D. Estrin, Data-centric storage in sensornets, ACM SIGCOMM Computer Communications Review 33 (1) (2003) 137–142.

[17] A. Silberstein, R. Braynard, J. Yang, Constraint-chaining: on energy-efficient continuous monitoring in sensor networks, in: Proceedings of the 25th ACM SIGMOD International Conference on Management of Data (SIGMOD'06), Chicago, Illinois, USA, June 2006, pp. 157–168.

[18] B. Xiao, H. Chen, S. Zhou, Distributed localization using a moving beacon in wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems (TPDS) 19 (5) (2008) 587–600.

[19] Y. Yao, J. Gehrke, Query processing for sensor networks, in: Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR'03), Asilomar, California, USA, January 2003.

[20] B. Yu, B. Xiao, Detecting selective forwarding attacks in wireless sensor networks, in: Proceedings of the 20th International Parallel and Distributed Processing Symposium IPDPS 2006 (SSN2006), Greece, April 2006, pp. 1–8.