

A Cover-Up Scheme for Data-Centric Storage in Wireless Sensor Networks*

WEN-HWA LIAO, KUEI-PING SHIH[†] AND WAN-CHI WU

Department of Information Management

Tatung University

Taipei, 104 Taiwan

E-mail: whliao@ttu.edu.tw

[†]*Department of Computer Science and Information Engineering*

Tamkang University

Tamsui, 251 Taiwan

E-mail: kpshih@mail.tku.edu.tw

One critical problem in wireless sensor networks is how to make the effective use of the vast amount of data by providing sensor and sink nodes with efficient data storing and retrieving, respectively. Several data-centric storage techniques had been proposed to exploit data storing and retrieving, but all of them failed to effectively deal with hotspot storage due to irregular sensor deployments or data distribution. In this paper, we propose a cover-up scheme for data-centric storage in wireless sensor networks. In our scheme, the storage node generates a virtual coordinate to feign that it is far away from the hash location. A geographic routing protocol forwarding an event data to a storage node will ignore the original storage node and find a new storage node whose coordinate is the closest to the hash location. Our cover-up mechanism can adjust to another storage node dynamically when a storage node is full. Simulations have shown that our cover-up scheme can reduce the energy consumption and achieve better quality of data.

Keywords: data-centric storage, energy efficiency, hash location, load balancing, quality of data, wireless sensor networks

1. INTRODUCTION

The rapid advances in sensing and wireless communication technologies have facilitated the deployment of wireless sensor networks. A sensor network consists of a large number of tiny sensors with limited computation, memory, storage, communication range, and battery power [2, 4]. In typical sensor network applications, sensors are deployed to monitor one or more phenomena in an unattended environment, such as disaster management, habitat monitoring, industrial process control, and object tracking, *etc.* A vast number of the event data will be generated for a long sensing time in a wireless sensor network. Hence, it is critical to effectively use of the vast amount of data by providing sensor and sink nodes with efficient data storing and retrieving, respectively.

In [12], the storage policies can be classified into three categories for wireless sensor networks: local storage, external storage and data-centric storage. In local storage, the data is stored in the sensor node local memory when the event is detected. Since the

Received November 14, 2007; revised March 11, 2008; accepted April 3, 2008.

Communicated by Makoto Takizawa.

*A preliminary version of the paper has been published in IEEE Symposium on Computers and Communications (ISCC'07), Aveiro, Portugal, July 1-4, 2007, and was supported in part by Tatung University, under grant No. B96-N05-048.

sink node does not know which sensor nodes store the interested data, it typically executes the blind flooding over the whole wireless sensor network for sending a query packet that defines the criteria of its interests. The external storage proposes an alternative mechanism. Once a sensor node detects events, the data is stored at the external sink. Although there is no cost for sink query, it may waste so much energy to transmit to the sink node the data that it is not interested in. In the data-centric storage, there are some data-centric nodes selected from the wireless sensor network that are responsible for the data storing and retrieving. When an event is detected by a sensor node, the data is stored by name at the corresponding data-centric node. Since all sensor nodes and sink nodes are aware of the information of the data-centric nodes, they do not use blind flooding for sending data or queries to the data-centric nodes. Thus, data-centric can clearly conserve energy for data storing and retrieving in the wireless sensor network.

Both energy efficiency and load balancing of storage are critical considerations in the design of sensor networks. In [12], they proposed a data-centric storage based on a geographic hash table (GHT). GHT employs a structured replication to achieve load balancing. Structured replication uses a hierarchical decomposition of the geographic space which divides the network into multiple regions and creates an image point for each event in each region. When a node detects an event, it only sends the event data to the closest image point. Queries are routed to all image nodes, starting at the root and continuing through the hierarchy. There are several problems in the GHT. First, all the queries are concentrated on a storage node which is the closest to the hashing location. The storage node may suffer from hotspot storage problem. Second, the storage nodes are fixed and cannot adjust dynamically. If an event type occurs frequently, the storage nodes cannot store all of the event data. In this paper, we propose a load balancing scheme for data-centric storage. The scheme can achieve load balancing, avoid hotspot, and improve the data quality.

The rest of the paper is organized as follows. Section 2 discusses related work in data storage for wireless sensor networks. Our proposed cover-up scheme is presented in section 3. Experimental results are discussed in section 4. Section 5 concludes this paper.

2. RELATED WORK

There are several previous works for data storage in wireless sensor networks. In [12], they proposed a data-centric storage scheme that is referred to as GHT. The GHT is a geographic hash table system that hashes an event type into a geographic location, and stores the data of the event at the sensor node that is closest to the hash location. GHT also employs a structured replication to spread the load. A structured replication uses a hierarchical decomposition of the geographic space similar to that used in GLS [9] and creates multiple image nodes for each event. Given a root r and a hierarchy of depth d , it can compute $4^d - 1$ mirror images of r . A node detecting an event stores it at the mirror node, which is closest to its location. Queries must be routed to all mirror nodes and hence, structured replication reduces the cost of storage but increases the cost of queries. The GHT uses GPSR [8], a geographic routing algorithm to route events and queries to their corresponding nodes.

In [13], they adapted multiple hash locations to store data that makes each key dif-

ferent by combining its event type and resilient level. It makes the same event type with different locations and uses each hash location as the centroid of the geometric shape based on a mathematical calculation. Any node in this geometric area can be the storage node based on the residual power. However, the storage node must flood its new status to all the nodes. If an event type has k hash locations, the node will send k messages to those locations.

In [1], zone sharing (ZS) is a distributed scheme for the decomposition of storage hotspots. ZS presents the solution in the context of the distributed index for multi-dimensional data (DIM) scheme [10]. The benefit of the DIM's structure is that it stores data with similar attributes nearby and enables energy-efficient query resolution. When a hotspot arises in a set of sensor nodes, those nodes falling on the border of the hotspot will transfer some loads of their zones to some of their less-loaded neighbors. It will send information to notify its neighbors and increase the cost of communication. Although ZS can distribute the heavy loads of a sensor node to other nodes, it will incur extra overhead and destroy the DIM structure.

3. THE COVER-UP SCHEME FOR DATA-CENTRIC STORAGE

3.1 Basic Idea

The GHT uses a geographic hash table that maps event types to geographic locations. However, GHT uses a static hash function. These hash locations are mapped into the closest sensor nodes as storage nodes, so that the storage can get exhausted quickly. GHT cannot spread the load to another node unless there is a node which is closer to hash location than to the current storage node.

In our scheme, the storage node generates a virtual coordinate to feign that it is far away from the hash location. A geographic routing protocol forwarding an event data to a storage node will ignore the original storage node and find a new storage node whose coordinate is the closest to the hash location. The sensor node still keeps its original geographic location. It is not easy to predict the number of storage nodes for each event due to irregular data distribution. A key concept of the virtual coordinate of our scheme, which can solve the problem and adjust the storage node dynamically, is called a cover-up scheme.

3.2 System Model

In a wireless sensor network, we assume that nodes know their geographic locations by means of the GPS [2] or some other location service methods [6, 11]. We also assume that the network is a planar graph. On a planar graph, each node has one or more adjacent faces. For example, in Fig. 1, node a has five adjacent faces, A , B , C , D , and E . When a discovery message comes from one edge, the message is sent to the next edge. The coordinates of the nodes are added to the discovery message. After a discovery message finishes traversing on a face, all locations of the nodes are collected. The node sending the discovery message informs all the nodes on the face of the result. Hence, each node knows the positions of the nodes in the same face f . In our system model, we use this information to form a *Face_Node* table. Each node has a *Face_Node* table in the wireless

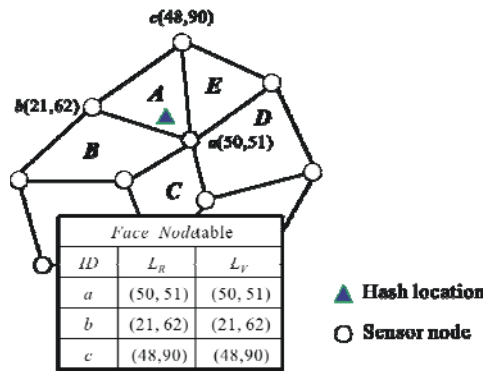


Fig. 1. After a discovery message finishes traversing face A, nodes a , b , and c , on the face A have the same *Face_Node* table.

sensor networks. Fig. 1 shows an example of the *Face_Node* table, which consists of ID , L_R , and L_V denoting the identification, the realistic coordinate, and the virtual coordinate of each sensor node, respectively. In the *Face_Node* table, we set the initial $L_V = L_R$. All nodes in the same face have the same *Face_Node* table.

There are three types of packets in our system model. First, when a node storage is full, the node will send a *Storage_Full* packet to notify all nodes on the same face. A *Storage_Full* packet consists of ID and L_V . Second, when a node detects an event, it will contain the event data and send *Put* packet to the storage node. A *Put* packet consists of ET and ED denoting event type and event data, respectively. Third, when a sink wants to retrieve the data of an event, it will send *Get* packet to the storage node. A *Get* packet consists of ET .

3.3 Store the Events

In Fig. 2, a node uses a hash function to generate a hash location L while the node detects an event. The node sends a *Put* packet by GPSR perimeter routing to identify a storage node, which is closest to L . The *Put* packet traverses the entire perimeter that encloses the destination L (e.g., the perimeter abc). According to the *Face_Node* table, node $a(50, 51)$ is the closest to $L(47, 53)$ and serves as the storage node. When a node detects the same event type, it will send the data to the same node until the node storage is full.

In GHT, if node storage is full, it will drop the coming data. We propose a cover-up scheme to solve the quality of data (QoD) problem. If a node storage is full, it will set its $L_V = (\infty, \infty)$ to cover-up the original coordinate in its *Face_Node* table, called cover-up node. Next, this storage node will send a *Storage_Full* packet to the nodes on the same face. A node receiving a *Storage_Full* packet will modify the $L_V = \text{Storage_Full}$ packet's L_V in its *Face_Node* table according to the *Storage_Full.ID*. Each node on the face, which encloses the hash location L , receives a *Put* packet to compute its L_V and neighbors' L_V on the same face which is the closest destination of *Put* packet. If the node's L_V is not the closest node to the destination of *Put* packet, it will deliver the packet to its neighbor, which is closest to the destination of *Put* packet. The procedure will stop until there is no node closer to the destination of *Put* packet than itself. The node, which is

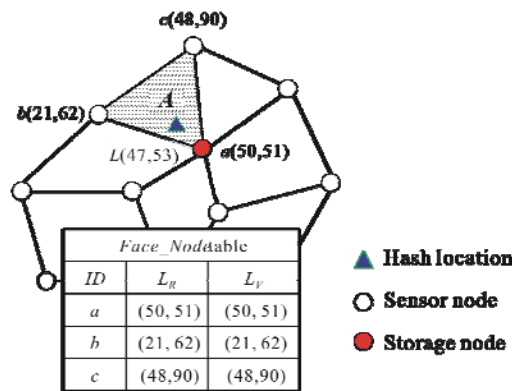


Fig. 2. Node a serves as the storage node since it is the closest to $L(47, 53)$.

```

Algorithm Cover-up Store
Variable:
ET: event type; ID: node's id;  $L_R$ : realistic location;  $L_V$ : virtual location;
L: the hash location of ET;
begin
for all sensors  $s_i$  on the face
while  $s_i$  receives Put packet do
    if ( $L_V$  of  $s_j$  is the nearest L in Face_Node table of  $s_i$ )
     $s_j$  stores the event
    endif
if (storage of  $s_i \geq$  threshold)
     $L_V$  of  $s_j = (\infty, \infty)$ 
     $s_j$  sends Storage_Full packet to all nodes on the face
    endif
endwhile
    while  $s_i$  receives Storage_Full packet do
    update  $L_V$  of Storage_Full.ID of Face_Node table
    endwhile
end
    
```

Fig. 3. The cover-up store algorithm.

closest to the destination stores the event data and becomes the new storage node. The old storage node doesn't store the event data because its L_V is farther than the new storage node to the destination of *Put* packet. In our cover-up scheme, a closer node will become the new storage node and store the event data to avoid dropping data. Since the old storage node has two sets of coordinates, a realistic coordinate and a virtual coordinate, the latter is set to (∞, ∞) to avoid the node becoming the storage node again. Note that the realistic coordinate L_R delivers the packet of any type to the neighbors. The algorithm of storing the events is shown in Fig. 3.

Let us illustrate an example to show how the cover-up scheme works in detail. In Fig. 4, suppose that node a 's storage is full. First, node a sets its $L_V = (\infty, \infty)$ to cover-up the original coordinate $a(50, 51)$. Next, node a will send a *Storage_Full* packet to all the nodes (node b and node c) on the same face. Upon receiving the *Storage_Full* packet,

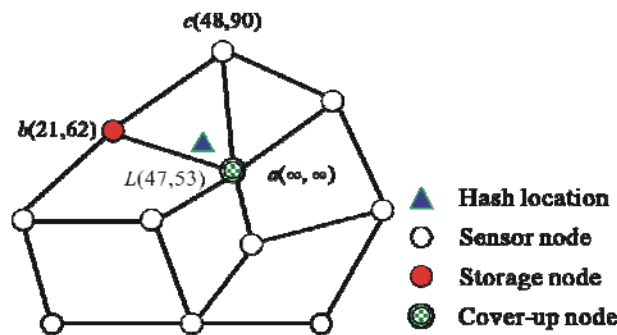


Fig. 4. When the node a 's storage is full, node a sets its virtual coordinate $L_V = (\infty, \infty)$ to cover-up the realistic coordinate $a(50, 51)$ and node b serves as the new storage node.

nodes b and c will update the L_V which is $ID = ID$ of *Storage_Full* packet of their *Face_Node* table. When a node detects an event again, it uses GPSR to route the *Put* packet to find the new storage node. When the *Put* packet is routed into face abc , the first node (assume node a) receiving the *Put* packet will use L_V to check its location with the destination of *Put* packet. Since node a 's L_V is (∞, ∞) , it is not the closest to the hash location L . Hence, node a does not serve as the storage node. According to the *Face_Node* table of node a , L_V of node b and L_V of node c are used by node a to compute the distances from node b and node c to the hash location L . Assume that the distance from node b to hash location L is the shortest. Node a sends the *Put* packet to node b according to the computed result of node a . When node b receives the *Put* packet, it repeats the checking procedure of the storage node. After checking the procedure of the storage node, node b is the closest to the hash location L and stores the *Put* packet. Hence, node b serves as the new storage node.

3.4 Retrieve the Events

Under our cover-up scheme, each node on the face which encloses the hash location L may store the event data for each event by adjusting L_V . Hence, any node on the face can possibly store the event data. When a sink wants to retrieve data, it sends a *Get* packet to all nodes on the face enclosing the hash location L . When a node receives the *Get* packet, it first checks if the *Get* packet was actually received or not. If not, then the node checks its event data if it corresponds to the *Get* packet or not. If the event data corresponds to the *Get.ET* packet, the node sends its event data to the sink. The node continues to send the *Get* packet to its neighbor. If a node receives the *Get* packet, it discards the *Get* packet and stops to forward the *Get* packet to its neighbors on the same face. The algorithm of retrieving the events is shown in Fig. 5.

Figs. 6 and 7 illustrate examples for data retrieval process. The sink forwards a *Get* packet to a face abc which encloses the hash location L . When the *Get* packet enters into face abc and node a receives the *Get* packet, it checks whether it actually received the *Get* packet. If node a did not receive the *Get* packet, then node a checks its event data to see whether they corresponds to the *Get.ET* packet. Node a stores the event data corresponding to the *Get* packet and then sends the event data back to the sink. Node a continues to send the *Get* packet to its neighbor. When nodes b and c receive the *Get* packet,

```

Algorithm Cover-up retrieve
Variable:
ET: event type; ID: node's id; LR: realistic location; LV: virtual location;
L: the hash location of ET;
begin
for all sensors si on the face
while si receives Get packet do
    if (Get packet is duplicated)
        Get packet is discarded
    else
        if (si has the data of Get.ET)
            si sends the data back to the sink
            si sends Get packet to next node on the face
        endif
    endif
endwhile
end
    
```

Fig. 5. The cover-up retrieve algorithm.

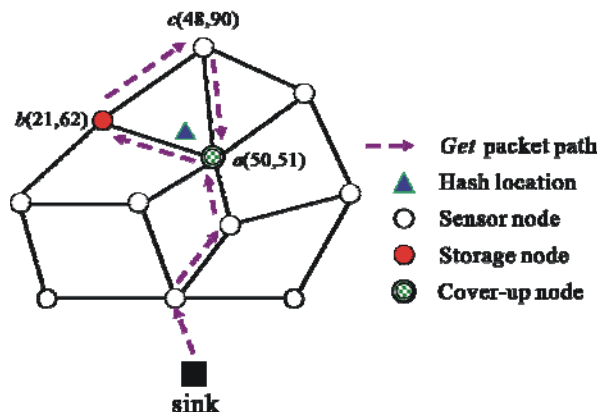


Fig. 6. Node *a* forwards the *Get* packet to its neighbors on the face *abc*.

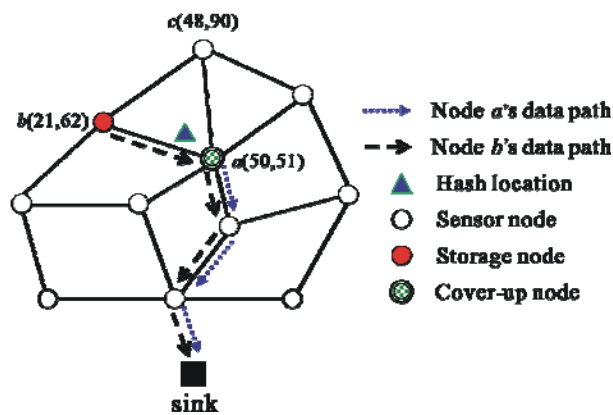


Fig. 7. Nodes *a* and *b* send their data back to the sink.

they repeat the procedure taken by node a . If node b has the event data corresponding to the *Get* packet, then node b will send the event data back to the sink. On the contrary, if node c does not have the event data corresponding to the *Get* packet, then node c will stop sending the *Get* packet.

4. SIMULATION EXPERIMENTS

In this section, the simulation evaluates the performance of our cover-up scheme and compares it with the GHT. In our system model, the sensor nodes can operate in one of two modes: sending message and receiving message. The total energy consumption includes the energy consumed in storing the events and the energy consumed in retrieving the events. We use an energy model [5] to estimate the energy consumption. The energy consumption for sending a message is determined by a cost function: $E_{send} = E_{trans} * s + E_{amp} * d^2$, where E_{trans} is the energy cost of sending a bit, s is the message size, E_{amp} is the energy consumed in the amplifier, and d is the distance of the message transmission. The energy consumption for receiving a message is determined by a cost function: $E_{receive} = E_{rec} * r$, where E_{rec} is the energy cost of receiving a bit, and r is the message size. The simulation parameters are given in Table 1. There is a single sink located at coordinate (0, 0) of the network which serves as the query source for all simulations.

Table 1. Simulation parameters.

Symbol	Definition	Setting
N	Number of sensor nodes	800
$X * Y$	Size of the sensing region	400m * 400m
R	Transmission range	30m
W	Number of event's types	4
D	Node's density	1 node/200m ²
E_{trans}	Transmitter electronics	50 nj/bit
E_{rec}	Receiver electronics	50 nj/bit
E_{amp}	Transmit amplifier	0.1 nj/bit/m ²
S_{memory}	Size of a node memory	15 events
S_{query}	Size of a query packet	4 bytes
E	Number of events	200
Q	Number of queries	200

First, we evaluate the energy consumption for data storage. Secondly, we evaluate the energy consumption for data retrieval. Thirdly, we evaluate the total energy consumption for data storage and data retrieval. Finally, we evaluate the number of dropped events for the QoD. The number of queries is constant while the number of the events is varied. The events occur randomly in the sensing area.

4.1 The Energy Consumption for Data Storage

Fig. 8 presents the energy consumption for data storage of the cover-up and GHT

with varying number of events. The result of the energy consumption for data storage shows that our cover-up scheme consumes less energy than the GHT with various number of the events. The reason is that the GHT employs a structured replication, so the event must be stored in many nodes.

4.2 The Energy Consumption for Data Retrieval

Fig. 9 presents the energy consumption for data retrieval of the cover-up and GHT with varying number of events. The result of the energy consumption for data retrieval shows that our cover-up scheme consumes less energy than the GHT with various number of the events. The reason is that the GHT must send a query to all of image points, and every corresponding sensor will deliver the data back to the sink. In the contrast, our cover-up only sends a query to the nodes on the same face. Hence, our cover-up consumes less energy than the GHT.

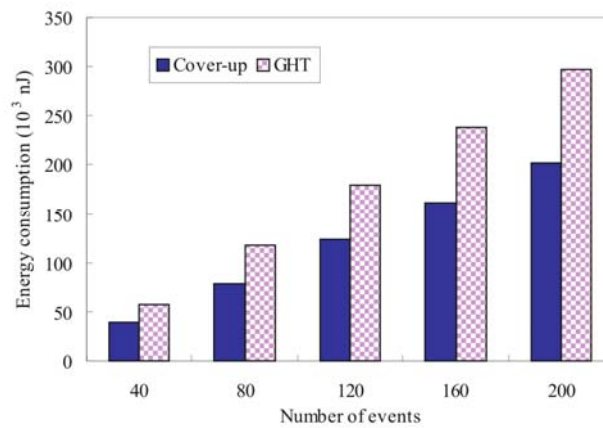


Fig. 8. The energy consumption with data storage.

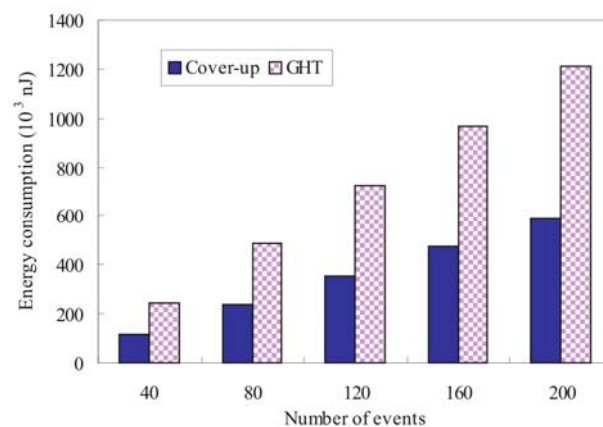
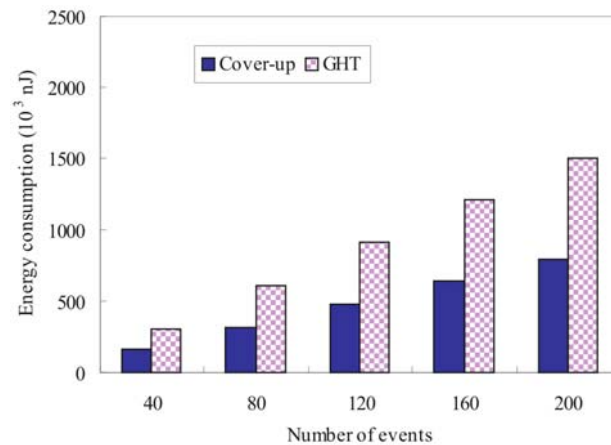


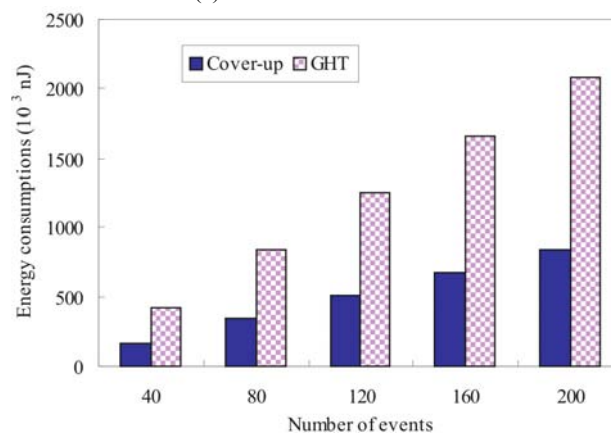
Fig. 9. The energy consumption with data retrieval.

4.3 The Energy Consumption for Data Storage and Data Retrieval

Fig. 10 presents the total energy consumption for data storage and data retrieval of the cover-up and GHT with varying number of events. In Figs. 10 (a) and (b), the network environment is $N = 800$ and $N = 1250$ over $X * Y = 400\text{m} * 400\text{m}$ and $X * Y = 500\text{m} * 500\text{m}$, respectively. The result of the total energy consumption for data storage and data retrieval shows that our cover-up scheme consumes less energy than the GHT with various number of the events. Hence, our cover-up scheme can significant reduce energy consumption for data-centric storage in wireless sensor networks.



(a) $X * Y = 400\text{m} * 400\text{m}$.



(b) $X * Y = 500\text{m} * 500\text{m}$.

Fig. 10. The energy consumption for data storage and retrieving.

4.4 The Quality of Data

Fig. 11 presents the QoD with varying number of events. In Fig. 11 (a), when the number of events varies from 40 to 120, there are no events dropped in the cover-up and

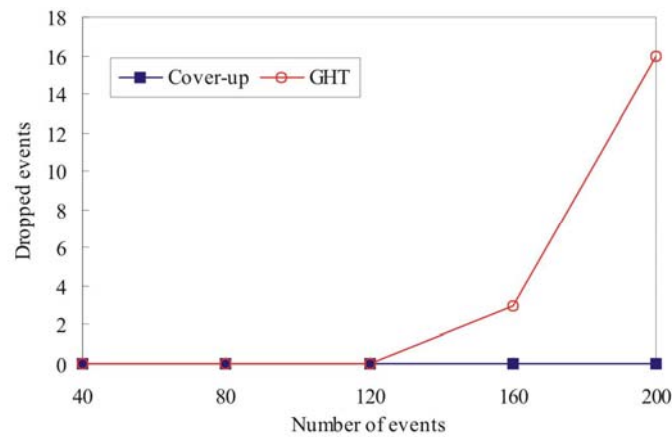
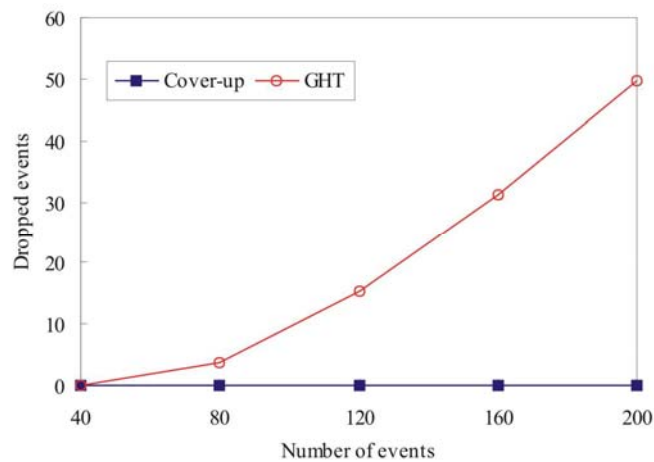
(a) $X * Y = 400 * 400$ and $R = 30m$.(b) $X * Y = 600 * 600$ and $R = 50m$.

Fig. 11. The quality of data with varying number of the events.

GHT. In the GHT, there are 5 to 15 events which are dropped due to storage space constraints and irregular data distribution when the number of events is higher than 120. In our schemes, the number of dropped events is almost equal to 0 in a sensor network with 40 to 200 events. In Fig. 11 (b), when the number of events are 40, there are no events dropped in the cover-up and GHT. In the GHT, there are 5 to 50 events which are dropped when the number of events is higher than 40. In our schemes, the number of dropped events is almost equal to 0 in a sensor network with 40 to 200 events. It is clear that our schemes can improve the QoD by dropping fewer events.

5. CONCLUSION

In this paper, we presented the cover-up scheme for decomposing storage load of hotspot storage in data-centric storage wireless sensor networks. The cover-up adjusts

the storage node dynamically and spreads the data into several storage nodes to obtain load balancing among the sensor nodes. We have shown experimentally that the applications our cover-up scheme is able to achieve good performances. The cover-up is able to adjust the storage nodes dynamically in order to reduce energy consumption and achieve better quality of data.

REFERENCES

1. M. Aly, N. Morsillo, P. K. Chrysanthis, and K. Pruhs, "Zone sharing: A hot-spots decomposition scheme for data-centric storage in sensor networks," in *Proceedings of International Conference on Very Large Data Bases Workshop on Data Management for Sensor Networks*, 2005, pp. 21-26.
2. T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Proceedings of Mediterranean Conference on Control and Automation*, 2005, pp. 719-724.
3. N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for wireless sensor networks," *IEEE Personal Communications Magazine*, Vol. 7, 2000, pp. 28-34.
4. D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer*, Vol. 37, 2004, pp. 41-49.
5. W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000, pp. 1-10.
6. J. Hightower, G. Boriello, and R. Want, "SpotON: An indoor 3D location sensing technology based on RF signal strength," Report #2000-02-02, Department of Computer Science Engineering, University of Washington, 2000.
7. Q. Huang, S. Bhattacharya, C. Lu, and G. C. Roman, "FAR: Face-aware routing for mobicast in large-scale sensor networks," *ACM Transactions on Sensor Networks*, Vol. 1, 2005, pp. 240-271.
8. B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*, 2000, pp. 243-254.
9. J. Li, J. Jannotti, D. DeCounto, D. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, 2000, pp. 120-130.
10. X. Li, Y. J. Kim, R. Govidan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proceedings of ACM Conference on Embedded Networked Sensor Systems*, 2003, pp. 63-75.
11. A. Nasipuri and K. Li, "A directionality based location discovery scheme for wireless sensor networks," in *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 105-111.
12. S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A geographic hash table for data-centric storage," in *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, pp. 78-87.
13. R. Tamishetty, L. H. Ngoh, and P. H. Keng, "An efficient resiliency scheme for data

centric storage in wireless sensor networks,” in *Proceedings of IEEE Vehicular Technology Conference*, Vol. 4, 2004, pp. 2936-2940.



Wen-Hwa Liao (廖文華) received the B.S. degree in Computer Science from Soochow University, Taiwan, in 1989, and the M.S. and Ph.D. degrees in Computer Science and Information Engineering from National Central University, Taiwan, in 1992 and 2002, respectively. From 2003 to 2004, he was an Assistant Professor at the Department of Information Management, Chang Jung Christian University, Taiwan. He joined the faculty of the Department of Information Management, Tatung University, Taiwan, as an Assistant Professor in 2004. From October of 2001 to May of 2002, he was a visiting scholar at the Center for Advanced Computer Studies, University of Louisiana at Lafayette, U.S. His current research interests include wireless sensor networks, mobile ad hoc networks, wireless networks, and mobile communications. He is a member of the IEEE Computer Society, the IEEE Communication Society, and the Phi Tau Phi Scholastic Honor Society.



Kuei-Ping Shih (石貴平) received the B.S. degree in Mathematics from Fu Jen Catholic University, Taiwan, in June 1991 and the Ph.D. degree in Computer Science and Information Engineering from the National Central University, Taiwan, in June 1998. After two years of military obligation, he joined the faculty of the Department of Computer Science and Information Engineering, Tamkang University, Taiwan, as an Assistant Professor in 2000. Currently, he is an Associate Professor in the Department of Computer Science and Information Engineering, Tamkang University, Taiwan. Dr. Shih's current research interests include network protocol designs for wireless LANs, MANs, wireless sensor networks, and mobile ad hoc networks. Dr. Shih has received the research award of National Science Council, Taiwan, 2000 and several best paper awards from conferences. He is a member of the IEEE Computer Society, the IEEE Communication Society, and the Phi Tau Phi Scholastic Honor Society.



Wan-Chi Wu (吳婉綺) received the B.S. degree in Information Management from Southern Taiwan University, Taiwan and the M.S. degree in Information Management from Tatung University, Taiwan, in 2004 and 2007, respectively. Her current research interests include wireless sensor networks and mobile ad hoc networks.