SDS: Distributed Spatial-Temporal Similarity Data Storage in Wireless Sensor Networks

Haiying Shen, Ting Li, Lianyu Zhao and Ze Li
Department of Computer Science and Computer Engineering
University of Arkansas, Fayetteville, AR 72701
Email: {hshen, txl005, lxz014, zxl008}@uark.edu

***Abstract*—Since centralized data storage and search schemes often lead to high overhead and latency, distributed data centric storage becomes a preferable approach in large-scale WSNs. However, most of existing methods lack optimization for spatial-temporal search and similarity search for multi-attribute data. Some methods are optimized under circumstances where nodes are equipped with locating systems (e.g., GPS) which consumes high energy. This paper proposes a distributed spatial-temporal similarity data storage scheme (SDS). It disseminates event data in such a way that the distance between WSN neighborhoods represents the similarity of data stored in them. In addition, SDS carpooling routing algorithm efficiently routes messages without the aid of a locating system. SDS provides efficient spatial-temporal and similarity data searching service. Experimental results show that SDS yields significant improvements on the efficiency of data querying compared with existing approaches.**

I. Introduction

A wireless sensor network (WSN) is a wireless network consisting of a large number of distributed low-power and inexpensive sensor devices. WSNs have been used in many military and civilian application areas such as military target tracking, habitat monitoring, health monitoring, environmental contaminant detection and industrial process control [1].

A data storage scheme is an indispensable component in WSNs, which offers data storing and searching services. In addition to energy efficiency demand, fast data searching is another requirement for data storage scheme. To overcome the unnecessary communication cost and the unbalanced energy consumption of centralized data storage method, distributed data storage approaches have been proposed (e.g. [2] [3]). Most distributed data storage schemes hash event data to locations according to data names. Though these schemes enhance the speed of data searching, most of existing schemes lack optimization for spatial-temporal searching and similarity searching for multi-attribute data. In addition, some schemes are optimized under circumstances where nodes are equipped with locating systems [4] (e.g., GPS), which leads to high energy consumption. Spatial-temporal searching allows users to search data of events occurred in a specified physical location and time period. Similarity searching enables users to search similar data in a query rather than the exact data. Efficiently achieving these two functions still remains a crucial problem in WSNs, especially in large-scale distributed WSNs with a tremendous volume of data. For example, a human operator may pose a question: "how many pedestrians are there in the geographical region X during 7:00pm-8:00pm, March 4, 2009?". If the operator receives data for the entire area covered by a large-scale WSN during all the time, the latency to process the received data may lead to a delay to a time-critical military task. Similarity searching for querying data in a certain similarity degree provides more flexibility. It is often more appropriate for a user to formulate search requests in less precise terms, rather than defining a sharp limit.

In spite of the efforts for the deployment of WSNs, there has been very little research devoted to tackling efficient spatial-temporal similarity data storage in a large-scale WSN. This paper proposes a distributed spatial-temporal similarity data storage scheme (SDS) that accelerates querying speed and reduces communication energy consumption and overhead. Compared to other distributed data storage schemes, SDS is advanced in that it optimizes data querying based on not only data name but also data similarity, at the same time it offers spatial-temporal data searching. In addition, SDS does not need GPS to locate the positions of nodes for routing, while achieving comparable efficiency to GPS-based geographical routing. Thus, SDS reduces energy consumption and enhances data searching flexibility in WSNs.

The remainder of this paper is organized as below. In Section II, we describe and analyze related data storage approaches in WSNs. In Section III, we present the design of the SDS scheme. The performance evaluation of SDS is presented in Section IV. Conclusion and future work are given in Section V.

II. Related Work

Centralized storage. Centralized storage methods [5] store all of the data generated by sensor nodes to a single sink which locates outside of the WSN and processes data. A data query needs to travel to the sink to find the data source, thus the centralized sink may become a hot-spot. In addition, the energy cost is not distributed in balance. The energy consumption of the nodes closer to the sink is greater than nodes far away from the sink, because the latter needs the former to forward sensed data or queries to the sink. Therefore, centralized storage methods could only be used in small-scale WSNs and low data generation rate.

Local storage. In local storage schemes, all sensed data is stored locally at the sensor nodes that detected the data. Hence, there is no communication cost for storing sensed data.

However, because data can reside anywhere in the network, a data query must be flooded to all sensor nodes in the network, leading to high overhead and energy cost. Directed Diffusion protocol [6] is developed to save the cost in data querying. In the protocol, nodes disseminate interest messages throughout the sensor network. The events matching an interest flow towards the interested nodes along multiple paths. Then, an interested node determines routes for future data flow by choosing a high-quality path from each source node.

Zhang et al. [7] proposed an index-based data dissemination scheme in which sensed data is stored at the detecting nodes or close nodes, and the data location information is pushed to a number of index nodes. It avoids unnecessary data transmission and control message flooding over the entire network. TAG [8] provides SQL-like semantics. It uses a delivery tree to distribute operators to nodes, and uses aggregation method to gather query results from leaves to root.

Distributed storage. Recent research focuses on distributed data storage schemes for WSNs. These techniques differ in the aggregation mechanisms used, but are loosely based on the idea of geographic hashing. One such data storage scheme is Geographic Hash Table (GHT) that provides a hash function for mapping event data to locations [4]. GHT hashes a data name to a key first, and then decides where the data should be stored based on the key. The data having the same name will be saved at the same location. It uses geographical routing for locating data. Distributed Index of Features (DIFS) [9] and Distributed Multi-dimensional Range Queries (DIM) [10] extend the GHT approach to provide distributed hierarchies of indexes to data. DIM is geared towards multi-dimensional range queries, in which multi-attribute data is mapped to a k-bit binary vector, each of the 2^k possible binary codes is mapped to a unique zone in the network area. DIFS focuses on semantically rich high-level events. It allows range queries and efficient index construction using a spatially distributed index. DIMENSIONS [11], [12] incorporates long-term storage, which progressively discards old data while preserving its key features for future mining. It provides multi-dimension data access and sufficiently accurate responses to queries with low communication overhead [13].

To improve routing performance, GLS [14] arranges each mobile node to periodically update a small set of location servers with its current location. When a node queries for the locations of other nodes, it uses predefined identifier and spatial hierarchy to find a location server for those nodes. GEM [15] embeds a labelled graph to the topology of network to enable nodes to perform efficient routing by merely knowing the labels of its neighbors. GRAB [16] forwards data along interleaved mesh to a receiver. It also controls bandwidth using the credits of data messages, thus allowing sender to adjust the reliability of data delivery. TSAR [17] has two tiers: proxy tier and sensor tier. At the proxy tier, it uses multi-resolution index structure. At the sensor tier, it supports adaptive summarization that trades off energy cost against overhead. A cover-up scheme [18] incorporates an overlay and uses virtual coordinate to redirect storage from overloaded nodes to less hot ones. Since the point-to-point DCS is difficult to deploy, pathDCS [19] was proposed to provide an approach that only needs the construction of a standard tree and uses tree-based communication primitives. Aly *et al.* [20] noticed the importance of uniform data distribution, and proposed KDDCS to use K-D tree to maintain even storage while avoiding hot spots.

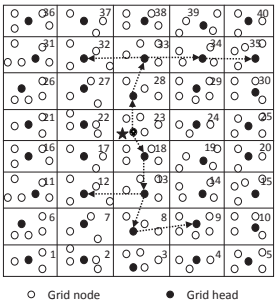


Figure 1. SDS zones and data transmission.

III. The Design of SDS

A. Design Goals and Strategies

(1) Similarity searching functionality. This functionality is very useful for collecting data that has inner relationship, SDS relies on locality sensitive hashing [21] to map data to zones in a locality-preserving manner as shown in Figure 1. That is, the physical closeness of zones represents the similarity of data in the zones.

(2) Spatial-temporal searching functionality. This makes it possible to search data of events occurred in specified locations and time periods. SDS tackles this challenge by building a two-dimensional space in each zone, and maps data to the space based on location and time.

(3) Low overhead. Our carpooling routing algorithm enables messages to travel together. Moreover, unlike GHT that depends on GPS for routing, SDS routing leads to comparable routing performance with no geographical knowledge available.

(4) Low latency. Zone-based routing algorithm in SDS takes zone rather than node as a routing step unit, thus accelerating message transmission. Further, it reduces congestion due to many messages as in Directed Diffusion. In addition, unlike GHT that searches each attribute in a query and then merges final results, SDS does not lead to long latency for processing located data.

B. Data Processing and Mapping

We consider a large-scale WSN that is deployed in a vast field, in which sensors are disseminated randomly. Without the loss of generality, we assume that the field is rectangle. SDS can also be extended to other shape of the WSN field. As shown in Figure 1, SDS divides the field horizontally and vertically to rectangular zones. The field is divided in such a way that nodes in any two adjacent zones can communicate with each other directly. That is, they are within each other's transmission range. IDs are assigned to successive zones in a sequential order. Thus, neighboring zones in the same horizontal level have successive IDs.

Each zone has a head which functions as the server for all other nodes (i.e. clients) in the zone. A zone head is mainly responsible for message routing, data assigning, and query responding for the nodes in the zone. A head maintains the information of the nodes within its zone, and the heads in its neighboring zones. It periodically exchanges "hello" messages with its clients and neighboring zone heads. To save energy consumption, in each zone, all nodes except the head are in sleep mode. Nodes in sleep mode still can sense data, but rely on the head for other functions. The nodes rotate to wake up to act as the zone head in a round-robin manner in order to balance the workload and energy consumption among nodes. Before transferring duty and corresponding information to a new head, the old head notifies its clients and neighboring heads of the new head.

Each sensor in the WSN has an identifier which can be the consistent hash value [22] of its IP address. After sensing an event data, a sensor processes the data, maps the data to nodes in a number of zones, and stores the data to the nodes. To process sensed data, a sensor derives the keywords of the data by strategies that allow contents of a file to be described with metadata [23]–[25]. For example, keywords *five car north* are derived from data *five cars are moving towards north*. We use d to represent the keywords of a sensed data item. A sensed data item is represented by a descriptor $< d, ID, t, s, (x, y) >$, where ID is the ID of the zone in which the data is sensed, t is the time when the data is sensed, s is the identifier of the sensor which sensed the data, and (x, y) is the exact location where the event occurred. Similarity of data item d_2 to data item d_1 is calculated by:

$$Similarity = \frac{|d_1 \bigcap d_2|}{|d_1|} \qquad (1),$$

where $|d|$ is the number of keyword in d. For example, the similarity of data item $Peter|CSCE|2456983|Arkansas$ to $Mary|CSCE|2468972|Arkansas$ is $2/4 = 0.5$.

For data mapping, SDS resorts to a locality-sensitive hashing function (LSH) [21] to transform d to a series of hash values. Data items having common keywords will have the same hash values, and similar data items will have close hash values. The number of hash values of a data item can be flexibly set in LSH. Higher value leads to fine-grained data clustering while lower value leads to coarse-grained data clustering. Taking one resultant hash value as an example, if the difference between d_1, d_2 and d_3 is $d_1 > d_2 > d_3$, their hash values conform to $h_{d_1} > h_{d_2} > h_{d_3}$, where h_d is the hash value of d. A detailed description of the LSH approach can be found in [21].

In the mapping between data and zones, the ID difference between zones indicates the similarity between the data stored in the zones. To achieve this objective, a data item with hash value h is mapped to the first zone with $ID \geq h$. For example, a data item with $h = 5$ will be stored in zone 5. If all nodes in zone 5 fail, the data will be stored in zone 6. Thus, we can see that the distance between neighboring zones in one horizontal level indicates the similarity of data in the zones.

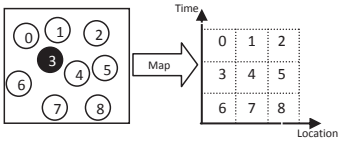


Figure 2. Spatial-temporal data mapping.

Within a zone, the data is further distributed among nodes according to their time and location. Figure 2 shows the data mapping within a zone in a spatial-temporal two-dimensional manner. Given a zone containing N nodes, the zone head calculates $k = \lfloor \sqrt{N} \rfloor$. The identifiers of nodes in the zone are normalized to identifiers from 1 to N. The head virtually arranges the nodes into a $k \times k$ grid as shown in the figure. Assume there are m zones in the WSN in total. The head divides range $[1, m]$ to k parts evenly. Taking a certain period of time, say one month, as a unit of time interval T, the head divides T to k parts evenly. During T, for a node with identifier i, it is responsible for the data of events occurred during the $(\lfloor i/k \rfloor + 1)^{th}$ time interval part and at the $(i\%k+1)^{th}$ location part. For example, in Figure 2, there are 9 nodes with identifier 0-8 within a zone, then $\lfloor \sqrt{N} \rfloor = 3$. For node 4, $\lfloor 4/3 \rfloor + 1 = 2$, $4 \bmod 3 + 1 = 2$. Thus, it is responsible for the second $\frac{1}{3}$ of the total time interval during T, and the second $\frac{1}{3}$ of the total range of location. In other words, a data item with t and ID will be stored in the node located in line $t\%\frac{T}{k}$ and column $ID\%\frac{m}{k}$ in the two-dimensional space.

C. Data Storage and Routing Algorithm

A sensed data item should be sent to and stored in a node for subsequent data querying. This is performed by the carpooling routing algorithm in SDS. Recall that one data item has a series of hash values (say n values), and it is mapped to the nodes based on the hash values. Thus, one data item will be stored in n nodes. When a node senses an event, it calculates n hash values of the data using LSH, $(h_1, h_2, h_3 \cdots, h_n)$. It then sends the event data $< d, ID, t, s, (x, y) >$ to its zone head along with the n destinations. Therefore, n copies of data should be sent to destination zones with IDs equal to $h_i (1 \leq i \leq n)$ respectively. The head determines the next hop among its neighbor heads based on each hash value. Assume the IDs of its neighboring zones are $ID_j (1 \leq j \leq 4)$. The next hop for each hash value h_i is the head in the zone with $min|ID_j - h_i| (1 \leq j \leq 4)$. That is, the next hop is the neighboring head that is the closest to the destination zone. Data copies targeting to different destination zones but the same direction are very likely to have the same next hop. In order to save overhead, rather than sending n data to n nodes directly, the head only sends one copy to each different next hop indicating their destination zones. After the next hop receives the data, it chooses the next hop in the same manner as the original head to forward the data. This process is repeated until the data is forwarded to its destination zones.

Figure 1 presents an example of the SDS carpooling routing algorithm. When a node in zone 23 senses an event, it reports the event data to its head. The head processes the event data and calculates the hash values of the data: 9, 12, 13, 32 and 35. The zone head sends one copy of the event data in the up

direction and one copy in the down direction. After the head in zone 18 receives the data, it further forwards it to zone 13. Then, the head in zone 13 sends one copy to zone 12, and also sends one copy to zone 8, which further forwards the data to zone 9. In the up direction, after the head in zone 28 receives the data, it forwards the data to zone 33. The head of zone 33 further forwards one copy to zone 32 and 35 respectively. After a data item is routed to the head of its destination zone, the head assigns the data to the node responsible for it. Using the data assignment method described in Section III-B, the head identifies the destination node of the data based on the data's ID and t, and forwards the data to the node.

The operation of data querying is performed in the same process as data storing. For a query of $< d, ID, t, s, (x, y) >$, the requester uses LSH to retrieve n hash values of the data. Taking the hash values as the IDs of the destination zones, the query is forwarded to n heads, and then to n nodes which respond to the queried data relying on the routing algorithm. If a query does not have specification of time and location, the head in the destination zone responds with all the data stored in the zone. If a query only has time (or location) specification, the destination head forwards the query to nodes in one line (or column) in Figure 2.

When a node searches for similar data, it needs to specify similarity degree. Recall that the difference of zone IDs represents the similarity of data in the zone. Thus, if a requester wants to receive more data with less similarity, it can specify a range r. r indicates the zone range need to be searched during data retrieval. For instance, for a hash value h, the zones with $ID \in [h - r, h + r]$ are searched. In routing, the head in the destination zone h forwards the query to the heads in zones $h - 1$ and $h + 1$. These heads further forward the query to zone $h - 2$ and $h + 2$, and so on until the heads in zone $h - r$ and $h + r$ receive the query. After a node receives a query, it checks whether the data it stores meets the similarity requirements using Equation (1) and responds with the desired data. For instance, if a query has

$$\langle h_1, h_2, ..., h_n \rangle, similarity = 50\%, r = 1,$$

then zones with ID equals to $[h_i - 1, h_i + 1](1 \leq i \leq n)$ will be searched, and data with similarity no less than 50% to the query will be retrieved.

D. Dynamic Data Management

In a mobile or unstable WSN, a sensor node may fail, leave or join in the network. It is important to ensure that a node in a zone always has the data that should be stored in itself in order to guarantee successful data querying. Specifically, when a new sensor node joins in the network, it contacts the zone head to identify the zone it resides at, and obtains the data it is responsible from other nodes. If a sensor node moves, before it moves out of its current zone, it notifies its zone head. The head moves the node's data to other nodes in the zone based on the data assignment algorithm. After the node moves out of its current zone, it contacts all neighbors until it identifies a new zone and head. It then refreshes its database, and acquires the event data from other sensors in the new zone. To avoid the data loss due to node failure, a zone head can have a copy of all data in its zone. After a failed node restarts, the node requests the ID of the zone and the data it should store from the zone head. This dynamic data management mechanism helps to ensure that all nodes are aware of their ID and data, which provides guarantee for successful data searching.

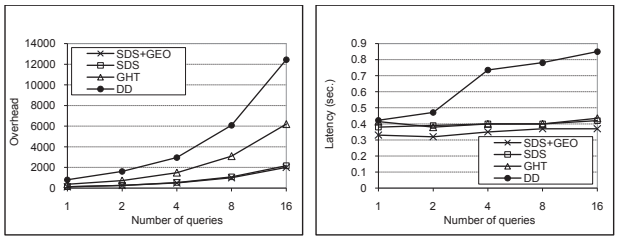
IV. Performance Evaluation

Our simulation is based The One simulator [26]. We evaluated the performance of SDS in comparison with Directed Diffusion (DD) [6] and GHT [4]. For a multi-attribute (i.e. keyword) query, DD uses broadcasting to search all desired data. GHT hashes each keyword for searching and merges the located data as the final results. We also include the results of SDS using geographical routing [4] rather than the ID-based carpooling routing algorithm, denoted by *SDS+GEO*. It routes data directly from a requester to destination node(s). The test scenario has a rectangular field of 400m$\times$400m, in which nodes are randomly and independently disseminated. Each node is aware of its own zone ID and its identifier within a zone. This field is divided into 4, 8, 16, 32, 40 data zones respectively, each of which has an area of $25m^2$ containing 25 nodes. Nodes generate event data randomly and send data at a speed of 100bps for 1400s. An event data item has size randomly chosen in [10,100] bytes and contains time and location together with ten different keywords. The number of hash values of a data item after the LSH operation is set to five. Before data querying operation, we gave each test 80s to distribute event data. All test queries are randomly generated at the speed of 10 queries per second. The number of nodes was set to 400 unless otherwise specified. We use the following metrics to test the performance of different methods.

(1) Overhead. It is the sum of the product of path length and message weight, which is the length of a message divided by the average message length. This metric measures the cost of querying and reflects energy consumption cost.
(2) Latency. It is the time period between a query is issued and response is received. Latency reflects the effectiveness of a method in quick data retrieval.
(3) Total number of events. It is the number of events that are returned in similarity queries. By comparing this number and the overall number of similar events, we can show the effectiveness of SDS similarity searching.
(4) Discovery rate. It is defined as the number of retrieved similar data items divided by all existing similar data items. This metric reflects the effectiveness of SDS in similarity data searching.

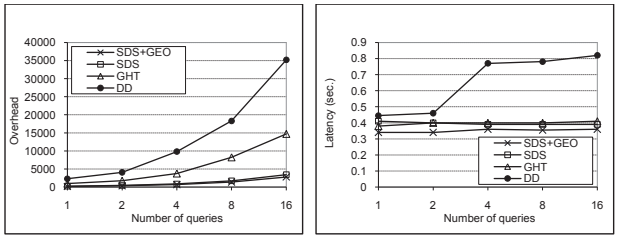
A. Spatial-Temporal and Range Querying

In the spatial-temporal querying experiment, we randomly generated 400 queries targeted at events that happened within a time interval of 2s as well as a location range of 20. Figure 3(a) shows the overhead of different methods versus the number of queries. We can observe that SDS and SDS+GEO generate the least overhead, DD generates the highest overhead, and GHT



(a) Overhead (b) Latency

Figure 3. Performance of spatial-temporal data search.



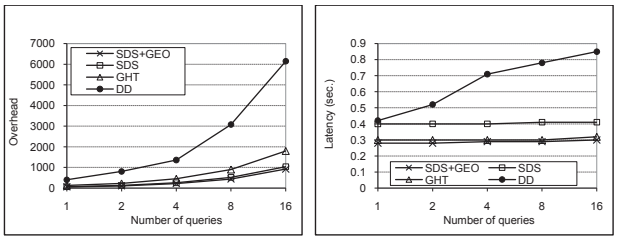
(a) Overhead (b) Latency

Figure 4. Performance of range querying.

locates in the middle. DD uses broadcasting for data querying, leading to significantly more messages, hence more overhead. For a query with ten keywords, a requester in GHT sends out one query for each keyword. After the requester receives response for each keyword, it merges the returned data for the results having the specified ten keywords. Further, it searches in the results for its desired data within the specified location and time period. Therefore, in GHT, much more data is routed back to the requester. In addition, GHT routing takes node as a step unit rather than a zone. As a result, it leads to much higher overhead than SDS. SDS always sends five queries and it only returns the desired data. Moreover, SDS routing takes zone as a step unit, leading to much less routing hops. We can also observe that SDS has comparable performance to SDS+GEO which takes the geographically shortest path with the aid of GPS. This result implies the efficiency the SDS carpooling routing algorithm. It forwards a message towards the direction of the destination between geographically close zones, generating almost the same overhead as SDS+GEO.

Figure 3(b) shows the latency of different methods versus the number of queries. We see DD has much higher latency than others. When the number of queries increases, the latency of DD grows sharply. SDS and GHT have almost the same latencies, and SDS+GEO leads to the least latency. As mentioned that GHT has much more returned data. Thus the increased traffic causes congestions, resulting in longer latency. ID-based routing in SDS without the aid of GPS should lead to longer latency since it may not take the shortest path. However, SDS has less routing traffic, which reduces the possibility of congestion. With geographical routing, SDS+GEO leads to less latency due to its less traffic and hops.

Range query is to find data items within certain range for similar results. The range was set to 3 in this experiment. Since GHT is not locality-preserving in data storage, its range querying cannot locate similar data. We still include its results for comparison. Figure 4(a) demonstrates the overhead versus the number of queries in range querying. We can observe that



(a) Overhead (b) Latency

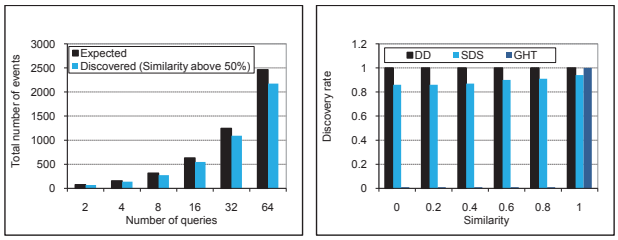
Figure 5. Performance of data searching.

DD produces dramatically higher overhead than others due to its broadcasting policy. Also, we find that GHT's overhead increases greatly as the number of queries increases. This is because GHT sends out six more queries for every attribute corresponding to the range size. SDS and SDS+GEO still generate the least overhead. In range querying, the destination zones only need to forward the queries to their neighboring zones, which generates a little more overhead. In addition, SDS routing takes zone as a step unit, leading to much less routing hops compared with GHT that uses node as step unit. The results show the range querying provided by SDS is efficient. Figure 4(b) shows that DD's latency increases dramatically due to congestion as the traffic grows. GHT and SDS have similar latencies. GHT takes the shortest path, but its latency is affected by the congestion caused by increased traffic. SDS does not need to send as many queries as GHT since it relies on neighboring forwarding. Thus, it has less traffic and reduces the possibility of congestion. SDS+GEO has the least latency because of less traffic of SDS and short routing path of geographical routing.

B. Single-Result Querying Performance

In this test, each query returns one data item in order to compare the performance of each approach without the influence of congestion due to returned data. The queries do not have time and location specification. Figure 5(a) shows the overhead versus the number of queries. We can see that SDS+GEO and SDS produce the least overhead, while DD performs the worst. The overhead of DD grows sharply as the number of queries increases. This is because DD broadcasts query messages to nodes, which results in more overhead. GHT generates more overhead than SDS. SDS sends a query to five zones as a result of LSH operation. It forwards a query along zones, and returns desired data to the requester. In addition, carpooling routing algorithm further reduces the overhead. GHT sends a query to ten destination nodes. It forwards a query along nodes rather than zones. Therefore, GHT leads to high overhead. We can also observe that the overhead of SDS is comparable to SDS+GEO. This confirms the efficiency of the SDS carpooling routing algorithm.

Figure 5(b) shows the latency versus the number of queries. We observe that DD leads to the longest latency, and GHT has less latency than SDS but performs slightly worse than SDS+GEO. As the number of queries increases, DD generates much more messages, which leads to congestion and longer latency. Without GPS, SDS uses ID-based routing algorithm that may not take the shortest path. Hence, its latency is not



(a) Number of discovered events (b) Discovery rate

Figure 6. Performance of similarity searching.

as short as GHT. SDS+GEO routes query directly to a node along the best path. Since SDS+GEO has less traffic than GHT due to less messages, it generates the least latency.

C. Similarity Searching

This test illustrates the effectiveness of SDS in similarity searching. We set the query range of SDS to $r = 3$. All queries have specification of 50% similarity. Figure 6(a) shows the number of discovered data items with similarity above 50% in SDS and the number of actual such data items in the system. We observe that SDS can always discover 90% of such data items. The results confirm that SDS is highly effective in similarity searching.

Figure 6(b) shows the discovery rate of each approach in term of similarity. Discovery rate means how many percent of data items that have a certain similarity to the query can be discovered. GHT can only return data with 100% similarity due to its exact match feature. Data in GHT is hashed to nodes using consistent hash functions. Thus, GHT fails to supply similarity searching which is very important in practice. Unlike GHT and SDS, nodes in DD broadcast queries to nodes. Consequently, it has 100% percent discovery rate for each similarity. However, its high discovery rate comes at the cost of high overhead. SDS provides an optimized trade-off between overhead and discovery rate of similar data. The results show that SDS obtains a discovery rate above 85% percent while avoids high overhead in DD.

V. Conclusion and Future Work

This paper proposes a distributed spatial-temporal similarity data storage scheme (SDS). Based on LSH, SDS efficiently disseminates data in a WSN such that similar event data is mapped into the same or nearby neighborhood(s), which enables SDS to offer similarity searching service. SDS also provides spatial-temporal data searching by classifying data into a two-dimensional space consisting of nodes in a neighborhood. Further, SDS carpooling routing algorithm can efficiently route queries or data without relying on GPS. The experimental results show that SDS not only provides better performance than Directed Diffusion and GHT in terms of overhead and flexibility, but also exhibits comparable latency to GHT which employs geographical routing. Our future work will be focused on improving the accuracy of similarity search and optimizing the performance of SDS in a mobile environment.

Acknowledgment

This research was supported in part by U.S. NSF grants CNS-0834592 and CNS-0832109.

References

[1] B. Krishnamachari, "networking wireless sensors," *Cambridge University Press, ISBN-10 0-521-83847-9*, 2005.

[2] Y. Yao, X. Tang, and E. peng Lim, "In-network processing of nearest neighbor queries for wireless sensor networks," in *Proc. of DASFAA06*, 2006.

[3] M. Chen, T. Kwon, Y. Yuan, and V. C. Leung, "Mobile Agent Based Wireless Sensor Networks," *JOURNAL OF COMPUTERS*, vol. 1, pp. 14–21, 2006.

[4] S. Ratnasamy, B. Karp, and D. Estrin, "GHT: A geographic hash table for data-centric storage." ACM Press, 2002, pp. 78–87.

[5] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a Sensor Network Expedition," in *Proc. of the 1st IEEE European Workshop on Wireless Sensor Networks and Applications*, 2004, pp. 307–322.

[6] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of MOBICOM2000*. ACM, 2000, pp. 56–67.

[7] W. Zhang, G. Cao, and T. L. Porta, "Data Dissemination with Ring-Based Index for Wireless Sensor Networks," in *Proc. of ICNP2003*, 2003, pp. 305–314.

[8] S. Madden, M. J. Franklin, and J. M. H. W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in *Proc. of OSDI*, 2002.

[9] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker, "DIFS: A distributed index for features in sensor networks," in *Proc. of First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[10] X. Li, Y. J. Kim, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proc. of SenSys03*. ACM Press, 2003, pp. 63–75.

[11] D. Ganesan, "DIMENSIONS: Why do we need a new data handling architecture for sensor networks," in *Proc. of the ACM HotNets*. ACM, 2002, pp. 143–148.

[12] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann, "An evaluation of multi-resolution storage for sensor networks," in *Proc. of SenSys03*. ACM Press, 2003, pp. 89–102.

[13] D. Ganesan, A. Cerpa, Y. Yu, D. Estrin, W. Ye, and J. Zhao, "Networking issues in wireless sensor networks," *JPDC*, vol. 64, pp. 799–814, 2004.

[14] J. Li, J. Jannotti, D. S. J. De, C. David, R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *Proc. of MobiCom2000*, 2000, pp. 120–130.

[15] J. Newsome and D. Song, "GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information," in *Proc. of SenSys03*. ACM Press, 2003, pp. 76–88.

[16] F. Y. G. Zhong, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks," *WINET*, vol. 11, pp. 285–298, 2005.

[17] P. Desnoyers, D. Ganesan, and P. Shenoy, "Tsar: A two tier sensor storage architecture using interval skip graphs," in *Proc. of SenSys05*. ACM Press, 2005, pp. 39–50.

[18] W.-H. Liao and W.-C. Wu, "A cover-up scheme for data-centric storage in wireless sensor networks," in *Proc. of ISCC2007*, 2007, pp. 963–968.

[19] C. T. Ee and S. Ratnasamy, "Practical data-centric storage," in *Proc. of NSDI*, 2006.

[20] M. Aly, K. Pruhs, and P. K. Chrysanthis, "KDDCS: A load-balanced in-network data-centric storage scheme in sensor network," in *Proc. of CIKM*, 2006, pp. 317–326.

[21] H. Shen, T. Li, and T. Schweiger, "An Efficient Similarity Searching Scheme Based on Locality Sensitive Hashing," in *Proc. of ICDT2008*, 2008.

[22] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web," in *Proc. of STOC97*, 1997, pp. 654–663.

[23] W. Nejdl, W. Siberski, M. Wolpers, and C. Schmnitz, "Routing and clustering in schema-based super peer networks," in *Proc. of IPTPS03*, 2003.

[24] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu, "Data management for peer-to-peer computing: A vision," in *Proc. of the 5th International Workshop on the Web and Databases*, 2002.

[25] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov, "Piazza: Data management infrastructure for semantic web applications," 2003.

[26] "http://www.netlab.tkk.fi/tutkimus/dtn/theone/."