

Distributed Indexing and Data Dissemination in Large Scale Wireless Sensor Networks

Yiwei Wu

Department of Computer Science
Georgia State University
Email: wyw@cs.gsu.edu

Yingshu Li

Department of Computer Science
Georgia State University
Email: yli@cs.gsu.edu

Abstract—Many data dissemination techniques have been proposed for wireless sensor networks to facilitate data dissemination and query processing. However, these techniques may not work well in a large scale sensor network where a huge amount of sensing data are generated. In this paper, we propose an integrated distributed Connected dominating set Based Indexing (CBI) data dissemination scheme to support scalable handling of large amount of sensing data in large scale wireless sensor networks. Our CBI can minimize the use of limited network and computational resources while providing timely responses to queries. Moreover, our data dissemination framework ensures scalability and load balance. Analysis and simulations are conducted to evaluate the performance of our CBI scheme. The results show that the CBI scheme outperforms the external storage-based scheme, local storage-based scheme and the data-centric storage-based scheme in overall performance.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) composed of a set of static or mobile nodes, which dynamically form a temporary network without any existing or pre-defined network infrastructure, are now being widely used in many applications, including environment and habitat monitoring, traffic control, and *etc.* Since sensor nodes are tightly constrained in processing ability, storage capacity and energy, data query processing and dissemination in WSNs are very challenging due to these inherent characteristics.

Many data dissemination schemes [1], [2], [3], [4] and [5] have been proposed for WSNs. One kind of scheme is the external storage-based (ES) data dissemination. In this scheme, an external node such as a base station is used to collect and store sensing data. Although the query is processed only at the base station, a large amount of sensing data needs to be forwarded to the base station. Hence, it is very inefficient when the sensing data updates very frequently.

Another kind of scheme is local storage-based (LS), such as [2]. In this scheme, a node only needs to send data to the query node whenever it satisfies the query conditions. However, the query needs to be flooded into the whole network to find the source holding the data of interest. Therefore, considering a large scale WSN, the network-wide flooding may introduce significant traffic.

A data-centric storage-based (DCS) data dissemination scheme is proposed in [4]. In this scheme, the sensing data of one event type are stored at certain nodes within the

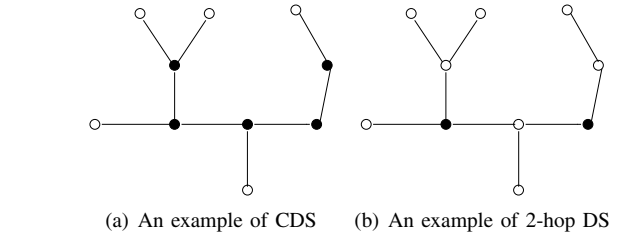


Fig. 1. Examples of CDS and k -hop DS, where black nodes represent dominators.

network which can be decided through a Geographic Hash Table (GHT). Although queries need not to be flooded, a large amount of sensing data still needs to be transmitted across the whole network to those storage nodes, which may introduce a huge communication overhead especially for large scale WSNs.

In order to provide low average query and storage communication and seek to balance those requirements over participating nodes, some work [6], [7], [8], [9] and [10] investigate constructing an index and its variations to facilitate query processing in WSNs. It seems that any kind of index on distributed data requires a hierarchical structure to aggregate information from the whole network. More detailed information can be accessed by a top-down traversal of the hierarchy to visit the sensors holding the relevant information. Some works in query processing and data dissemination for WSNs have presented a number of methods in constructing such hierarchies. However, most of those works adopt a tree-based index which is widely used in traditional databases. Since there does not exist any existing or pre-defined network infrastructure, it is difficult to maintain this tree-based index structure in WSNs. Hence, we adopt another important structure called Connected Dominating Set to form storage and index node sets. For a graph $G(V, E)$, a *Dominating Set* S of G is defined as a subset of V such that each node in $V \setminus S$ is adjacent to at least one node in S . A *Connected Dominating Set (CDS)* C of G is a dominating set of G which induces a connected subgraph of G . Fig. 1(a) shows an example of CDS. The nodes in C are called *dominators*, the others are called *dominatees*. A *k -hop dominating set* D in G is a set of nodes with the property that every node in G is at most k hops away from at least one of the nodes of D . Fig. 1(b) shows an example of 2-hop

dominating set. A CDS is the earliest structure proposed as a candidate for virtual backbones in WSNs. Having such a CDS simplifies routing by restricting the main routing tasks to the dominators only. Unlike in a traditional way where a node broadcasts the packet whenever it first receives one, all dominatees only need to send packets to their closest dominator(s) and only the dominators forward the packets towards their destinations. Some distributed methods [11], [12], [13] and [14] are proposed to construct a CDS in WSNs. Not only a CDS can be used to facilitate routing, it also can be used as data dissemination framework.

In this paper, we propose an integrated distributed Connected dominating set Based Indexing (CBI) data dissemination scheme to support scalable handling of large amount of sensing data in large scale WSNs. The main purpose of our CBI is to minimize the use of limited network and computational resources while providing timely responses to queries. Sensing data is collected and stored at the storage nodes close to the sensing nodes. This can decrease the communication cost significantly. Meanwhile, the information of high level semantically rich data are pushed and maintained at some index nodes. Hence, queries are only needed to be routed to the appropriate index nodes instead of flooding into the whole network. This data dissemination framework ensures scalability and load balancing of communication.

The remainder of this paper is organized as follows. In Section II, we review some existing data dissemination schemes. In Section III, our CBI data dissemination scheme is presented. In Section IV, we evaluate the performances of our CBI through simulations. Finally, we conclude this paper and discuss future research directions in Section V.

II. RELATED WORK

Only recently, how to construct a distributed index for a WSN to facilitate data dissemination has received some attention in the literature [3], [7], [8], [10], even though the index has been extensively studied in traditional database systems.

In [7], an approach DIFS which is based on GHT and quad tree was proposed to construct a distributed index in a WSN. DIFS builds on top of GHT which provides a keyvalue-based distributed index by hashing keys. DIFS constructs a multiply rooted hierarchical index which balances the overload from the root which is introduced by single quad tree approach to support efficient range queries. However, there are some drawbacks introduced by GHT and tree-based index. As we know, for GHT, every node should be aware of the boundary of the whole area. Although a tree-based index is very suitable in traditional databases, it still has some issues we should consider for WSNs, which is how to construct and maintain those index trees in a distributed way.

In [8], a spatio-temporal index structure, DIST, is proposed to trace mobile objects. In DIST, the whole sensor network is hierarchically decomposed into levels and there is a quad-tree like partitioning at each level. However, the repository nodes should be aware of the structure of the partition since it should

calculate the level of the plume and initialize index update. Therefore, this strategy will not work well if repository nodes are randomly selected or the whole network is very large.

In [3], a distributed index scheme for multidimensional data (DIM) which is based on the geographic routing (GPSR) is proposed to support range queries. To hash an event to a location, DIM uses a technique whereby events whose attribute values are “close” are likely to be stored at the same or nearby nodes. Then, GPSR is applied to route events and queries to their corresponding nodes in a distributed fashion. However, if the event types or attributes are not distributed well, DIMs scale very poor with network size.

In [5], a Two-Tier Data Dissemination approach was proposed. In their approach, every source builds a virtual grid structure to support efficient data dissemination for mobile sinks. Therefore, their approach is not suitable for dynamic sources. Otherwise, there is additional cost to maintain those virtual grid infrastructures along with the sources up and down. Although the query flooding can be confined within the region of around a single cell-size, four dissemination nodes of each cell would become immediate dissemination nodes due to the flooding nature, which renders this approach not efficient because of those duplicate data delivery.

The most related works with ours are [6]. In [6], an Adaptive Ring-based Index (ARI) scheme was proposed to facilitate data dissemination in large scale WSNs. In this scheme, the location information of those storing nodes is pushed to some index nodes, which act as the rendezvous points for sinks and sources. However, the index centers are determined by GHT on event type. Thus, this strategy requires each node to be aware of the boundary of the whole monitor area. Usually, it is difficult to pre-establish those boundaries for WSNs.

The main contribution of our work is that we propose an integrated distributed Connected dominating set Based Indexing (CBI) data dissemination scheme to support scalable handling of large amount of sensing data in large scale WSNs to overcome the drawbacks mentioned above. In our scheme, sensing data is collected and stored at the storage nodes, which form a k -hop dominating set of the whole network. Meanwhile, the information of high level semantically rich data are pushed and maintained at some index nodes formed by a connected m -hop dominating set. Hence, queries are only needed to be routed to the appropriate index nodes instead of flooding into the whole network. Our data dissemination framework can minimize the use of limited network and computational resources while providing timely responses to queries and ensures scalability and load balancing of communication.

III. CONNECTED DOMINATING BASED INDEX (CBI) SCHEME

In this section, we describe the basic idea of our data dissemination scheme. In this paper, we are mainly interested in static symmetric multi-hop wireless networks. The topology of a network is represented by a graph $G = (V, E)$, where V is the node set and E is the edge set. If two nodes are within the transmission range of each other, then there is an

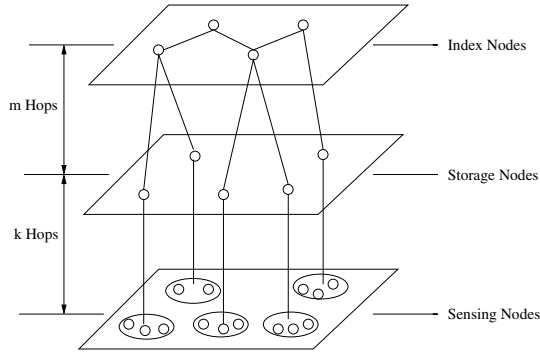


Fig. 2. Network Hierarchy

edge between them. We assume that all nodes are deployed in a 2-D plane. The area covered by a node is a circle with the radius equals this node's sensing range. There are many targets moving within a vast region. The sensor nodes in the network can detect the status of each target in its sensing range, and periodically generate sensing data. Users may issue a query via a sensor node (sink) for data about the current activities of each target. The result report also will be returned to the user via the sink.

In our scheme, the network is logically divided into three layers as Fig. 2 shows.

The bottom layer contains the sensing nodes that monitor the targets and generate raw sensing data.

The middle layer contains the storage nodes that are used to store the high level semantically rich data. The basic idea of how to choose storage nodes is that sensing data are collected and stored at the nodes close to the sensing nodes. Hence, in our work we construct a k -hop dominating set of the whole network and use it as a storage node set. Consequently, the maximum distance between the sensing nodes and storage nodes is at most k hops. Therefore, the raw data do not need to travel across the whole network. Thus, it can save limited network resource.

The top layer contains the index nodes that store the index information for those high level semantically rich data. The main design goal of a distributed indexing scheme is to provide timely and efficient response to queries while minimizing the amount of network and computational resources consumed by data dissemination. Hence, we use a connected m -hop dominating set as index node set to dominate the storage nodes only. According to the property of this connected m -hop dominating set, the maximum distance between storage nodes and the index nodes is at most m hops and the index node set is strictly connected as well. It is worth mentioning that although our scheme contains index nodes, we do not care about which data structure (e.g. quad-tree) should be used by a single index node. This strategy also gives our scheme more flexibility to process range and binary queries.

Obviously, storage nodes and index nodes in general consume more energy in handling various bypass traffic than sensing nodes. Those bypass overhead will drain the energy of those storage nodes and index nodes very quickly. In order to

prolong the life span of the whole network, hence, the storage nodes and index do not need to do or at most do some part of sensing task.

There are three cases we should consider when a query is injected into the network via a sink. If the sink is a sensing node in the bottom layer, this query will be forwarded to the storage node which dominates the sink firstly since the sink only knows which one is its storage node. From this storage node, the query will be forwarded to the index node which stores the index information for this storage node. Thus, the access time from the sink to the index node is a constant which is at most $k + m$ hops. If the sink is a storage node in the middle layer, the query will be forwarded to the index node directly. However, if the sink is an index node, the query only need to be processed at the top layer. Thus, no matter where is the sink, our scheme can provide timely and efficient response to queries if we choose sound values for k and m .

A. Storage Nodes Determination

According to the definition of k -hop dominating set, every node in G is at most k hops away from at least one of the nodes in a k -hop dominating set of G . Hence, we construct a k -hop dominating set of the whole network and all the nodes in this set are used as Storage Nodes. In this way, sensing data are collected and stored at the nodes close to the sensing nodes. Another advantage of using a k -hop dominating set as a storage node set is that storage nodes can combine data from different sources by using functions such as suppression (eliminating duplicates), Min, Max and Average, since sensor nodes might generate a significant amount of redundant data.

Firstly, we choose a *root* to start a breadth-first search (BFS). The root can be randomly selected or by using other leader election techniques. Then the root initializes a BFS search. After that, every node exchanges information with its k -hop neighbors about the level, degree and ID. After that all nodes should know all the k -hop neighbors' information. A node is called a *leaf node* if it has no children.

As we know, in order to obtain the storage node set with a small size and also make sure the subgraph composed by all the storage nodes has a small diameter, any leaf node should not be a storage node. Therefore, we construct a storage node set from bottom to top in the BFS tree. Before we present our procedure, we firstly introduce a three tuple variable which is $(Level, Degree, ID)$, where *Level* is the hop distance from the root after the BFS search, *Degree* is the number of the neighbors in the network and *ID* is the node's ID. We assume that each node has a unique ID. We say variable (L_1, D_1, ID_1) is larger than (L_2, D_2, ID_2) if $L_1 < L_2$ or $L_1 = L_2 \ \&\& \ D_1 > D_2$ or $L_1 = L_2 \ \&\& \ D_1 = D_2 \ \&\& \ ID_1 < ID_2$.

At first, the leaf node u with the smallest $(Level, Degree, ID)$ among its neighbors sends a DOMINATING message to its exact k -hop away parent v having the largest $(Level, Degree, ID)$ to request v to become a dominator (storage node). Whenever v receives this DOMINATING message, it becomes Black and broadcasts a BLACK message to all of its k -hop neighbors. Upon

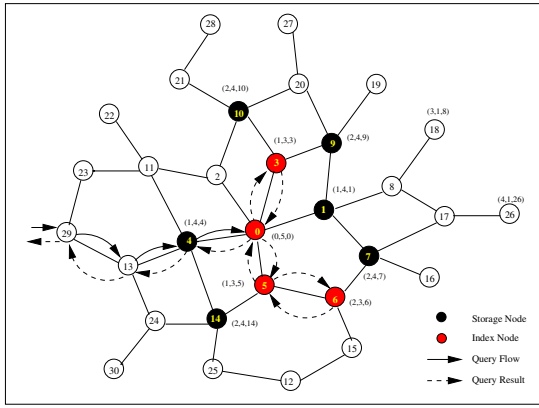


Fig. 3. An Example of Storage Nodes (Black) and Index Nodes (Red) Determination and Query process where $k = 2$ and $m = 1$.

receiving a BLACK message from its parent, u becomes Gray and broadcasts a GRAY message to all of its k -hop neighbors. Then, the next node with the smallest $(Level, Degree, ID)$ among its k -hop neighbors that have not decided their status yet starts this procedure in turn. It becomes Gray and broadcasts a GRAY message to its k -hop neighbors if it has received at least one BLACK message. Otherwise, it sends a DOMINATING message to its exact k -hop away parent with the largest $(Level, Degree, ID)$ as u does.

This procedure stops when the root becomes Gray or Black. After that, all Black nodes are used as Storage nodes that can k -hop dominate the whole graph as well.

Fig. 3 shows an example where $k = 2$ and $m = 1$, Node 26 is a leaf node with variable $(4, 1, 26)$. Node 26 sends a DOMINATING message to Node 7 since Node 7 has a largest $(Level, Deg, ID)$ within Node 26's 2-hop neighbors. Then Node 7 becomes Black and broadcasts a BLACK message. Similarly, Nodes 1, 4, 9, 10 and 14 will become Black. Therefore, in this particular case, no other nodes should become black and we use all those black nodes as storage nodes.

B. Index Nodes Determination

We construct a connected m -hop dominating set I to dominate all storage nodes only and use all the nodes in I as index nodes. Here, strict connectivity is required. As we know a data packet is much larger than an index packet. Therefore, it is desirable that $m > k$.

Firstly, we form a new graph G' which includes all the storage nodes and their parents in G . Then we construct a connected m -hop dominating set to dominate storage nodes only. However, all the storage nodes should be dominatees since they should not be index nodes again. Therefore, if the root becomes a storage node, we should pick up another node as the new one to start a new BFS search. The easiest way to find a new root is to transfer the role from the root to its white neighbor with the largest $(Level, Degree, ID)$, denoted as x . From x we build a BFS tree but restrict in G' only and also keep the storage nodes as leaf nodes as well.

After that we can use the same procedure used in the

storage node determination to find the dominators firstly. However, since we only need to dominate all the storage nodes, only a storage node should send a DOMINATING message to its exact m -hop away parent with the largest $(Level, Degree, ID)$.

Once we find all the dominators and color them Red, we should add some connectors to connect all those dominators, since in our work strict connectivity is required and we do not adopt the assumption that there exists a path between any pair of index nodes as other works do. In here, Each red node finds the shortest path to the root and marks all the internal nodes Red.

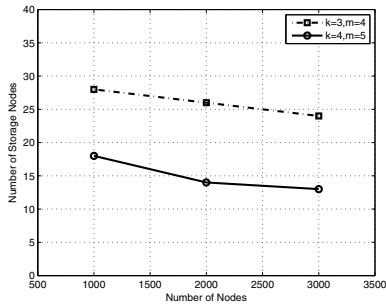
Finally, all the Red nodes form a connected m -hop dominating set which dominates all the storage nodes only. We now use all the nodes in this set as index nodes.

As shown in Fig 3, after we find all the storage (Black) nodes, we are going to determinate index nodes. We firstly build a BFS tree which restrictively only includes nodes 0, 1, 2, 3, 4, 5, 6, 7, 9, 10 and 14. All the storage nodes should be leaf nodes. However, a node's level may be different from the first BFS tree which is formed in the storage nodes determination phase. For instance, in order to keep Node 1 as a leaf node, Node 7 should change its parent from Node 1 to Node 6. Therefore, Nodes 0, 3, 5, 6 become Red and are used as index nodes.

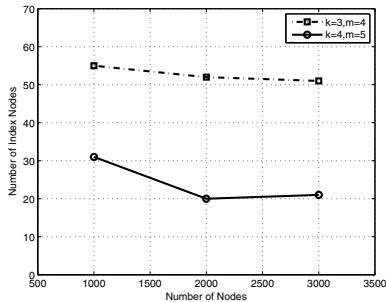
C. Index Construction

Most works use a quad-tree structure as a base to construct an index in a traditional database. Also, some works use the same idea to construct an index for a WSN and ignore the topology and routing details. However, for WSNs since nodes are randomly deployed in an area in most cases, it is not easy to divide the whole network into grids. Therefore, we use a simple method to construct an index for those high level semantically rich data. In this method, each index node only stores one copy index of its dominatees (storage nodes). We are not interested in any particular form of the index structure used in a single node, *e.g.* quad-tree, B-tree, *etc.* Actually, this scheme gives us more flexibility to process range and binary queries. Even though the query will be flooded to all the index nodes to get the query result, this flood overhead is much lower since the size of index node set is smaller enough compared with the total number of nodes in the whole network. In some cases when the query response time is the prime consideration, each index node also can broadcast its index to other index nodes whenever it is necessary, *e.g.* it detects data changes. In this way, the query does not need to visit all the index nodes and the access time is a constant which is at most $k + m$ hops. However this strategy has some extra overhead to update the index for all index nodes especially for a network with a large coverage level.

For example, when a query is injected into the sink (Node 29) in Fig. 3, this query is forwarded to the Storage Node 4 which stores the data for Node 29. Then, the query is flooded to all the index nodes. After that, the query result is returned to the sink after getting the result.



(a) Number of storage nodes



(b) Number of index nodes

Fig. 4. Comparison of storage nodes and index nodes with different k and m .

IV. SIMULATION

We conducted simulations to evaluate our scheme. In our simulations, nodes were placed uniformly at random within a 150×150 square. All nodes have the same transmission range 10 and sensing range 5. All of the data points were averaged over 50 simulation runs.

A. Comparison of the storage nodes and index nodes with different k and m

Fig. 4 plots the size of storage node set and index node set with different k and m by using our constructing method. From Fig. 4(a) we can observe that the size of storage node set decreases with the increasing of k and m . However, when the total number of nodes increases, the size of storage node set decreases since one storage node can dominate more sensing nodes when the network becomes dense. The size of index node set also expresses the same pattern shown in Fig. 4(b). One also can observe that the sizes of storage node set and index node set are much smaller than the total number of nodes in the whole network. This observation shows that our scheme has good performance in logically classifying the whole network to different layers.

B. Comparing performances of different data dissemination schemes

We compared the performance of CBI with other data dissemination schemes, ES, LS and DCS. Firstly, we define the *total message complexity* as the total number of messages generated in the whole network. The *hotspot message complexity*

is the maximum number of messages sent by one single node. The *total traffic complexity* is the amount of data sent by all nodes. The *hotspot traffic complexity* is the maximum amount of data sent by each node.

We choose $k = 3$ and $m = 5$ and the total number of nodes is 2000 in this simulation. Other parameters are listed as following. The size of a data message s_d is 80. The size of a query message s_q is 10. The size of an index update message s_i is 10. 10 mobile targets are randomly generated in the network. Each target randomly picks up a direction to move. It returns along the previous path whenever it moves out the boundary of the monitored area. A sensor node can detect the target whenever the target is in its sensing range. We also assume that the result of one target is returned for each query. The simulation duration time is 100.

Fig. 5 shows the message and traffic complexities of different data dissemination schemes when the target's velocity is 0.25. Based on this velocity, we can calculate the approximate average update ratio, which is $0.5 \left(\frac{0.25 \times 10}{5} \right)$. From the results, it is not surprising to see that no single scheme outperforms the others in all cases. From Fig. 5(a), although the total message complexity of CBI is a little bit larger than ES and DCS when event update rate/event query rate (r_u/r_q) is less than 1, it decreases significantly when r_u/r_q is larger than 1. From Fig. 5(b), the hotspot message complexity of CBI is larger when the r_u/r_q is larger than 2.5. However, our CBI is still very competitive with LS and DCS. Moreover, CBI has good performance in hotspot message complexity when r_u/r_q is smaller than 2.5 since the query is restricted only to the index nodes, the number of which is smaller enough compared with the total number of nodes in the whole network.

The total traffic of CBI shown in Fig. 5(c) is much lower than others because the property of our scheme is that sensing data are collected and stored at the nodes close to the sensing nodes and queries are restricted in a small number of nodes. We admit that the hotspot traffic complexity of CBI in Fig. 5(d) is larger than LS when r_u/r_q is larger than 5, since the query ratio is a dominating factor in LS. In fact, there is only a little difference between CBI and LS, and our CBI is still better than ES and DCS.

Finally, from those results and analyses, we can conclude that our CBI scheme has better overall performance than others in message and traffic complexities especially for r_u/r_q no larger than 5.

V. CONCLUSION

In this paper, we propose an integrated distributed Connected Dominating Based Indexing (CBI) data dissemination method to support scalable handling of large amount of sensing data in large scale wireless sensor networks. Our CBI can provide timely responses to queries while minimizing the use of limited network and computational resources. Our data dissemination framework ensures scalability and load balancing of communication as well as adaptivity in presence of dynamic changes. Analysis and simulation results show that

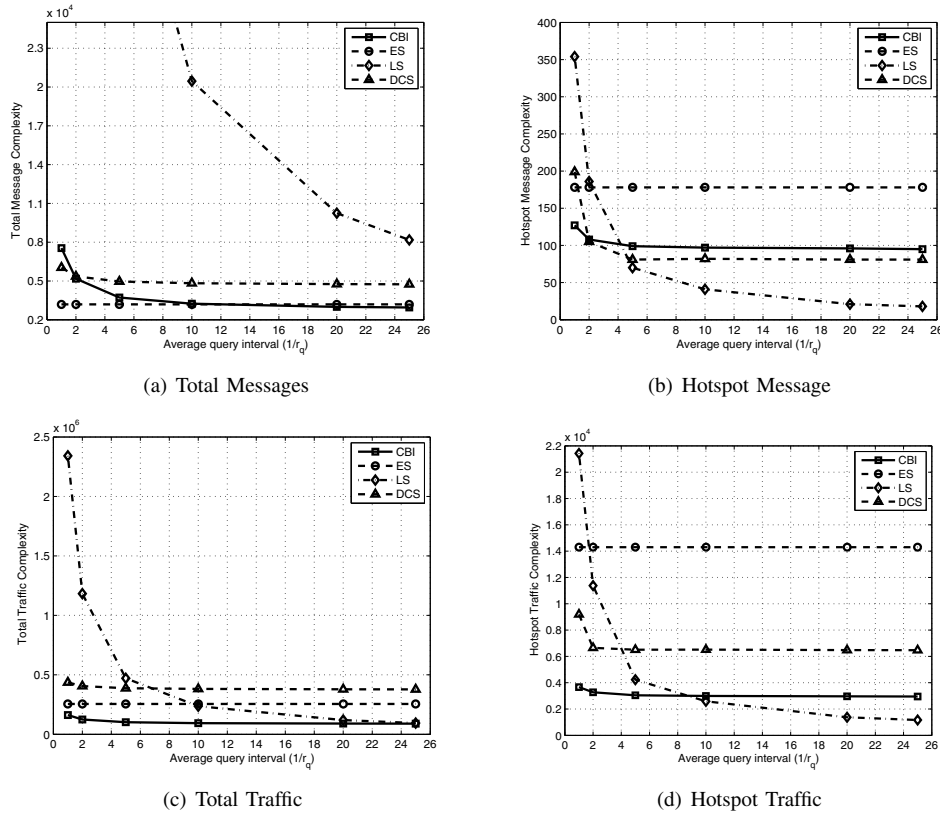


Fig. 5. Comparison of CBI, ES, LS, DCS when each target's velocity is 0.25.

the CBI scheme outperforms the ES, LS and the DCS schemes in overall performance.

Our future work is to investigate an efficient index data structure which can inherit the index structure such as quad-tree while keeping the properties of CDS as well. Another direction of future work is how to maintain our data dissemination framework in presence of network dynamic changes.

ACKNOWLEDGMENT

This work is supported by the NSF under grant No. CCF-0545667 and CCF 0844829.

REFERENCES

- [1] D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions: Why do we need a new data handling architecture for sensor networks?" in *Proceedings of the ACM Workshop on Hot Topics in Networks*, Princeton, NJ, October 2002, pp. 143–148.
- [2] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2–16, 2003.
- [3] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multi-dimensional range queries in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, Los Angeles, California, USA, 2003, pp. 63–75.
- [4] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensor networks with ght, a geographic hash table," *Mobile Networks and Applications*, vol. 8, pp. 427–442, 2003.
- [5] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "Ttd: Two-tier data dissemination in large-scale wireless sensor networks," *Wireless Networks Journal (WINET)*, vol. 11, pp. 161–175, 2005.
- [6] W. Zhang, G. Cao, and T. L. Porta, "Data dissemination with ring-based index for wireless sensor networks," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2003, pp. 305–314.
- [7] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and 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*, May 2003, pp. 163–173.
- [8] A. Meka and A. Singh, "Dist: A distributed spatio-temporal index structure for sensor networks," in *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, Bremen, Germany, 2005.
- [9] S.-Y. Park and H.-Y. Bae, "A distributed spatial index for time-efficient aggregation query processing in sensor networks," in *Proceedings of the International Conference on Computational Science (ICCS)*, 2005, pp. 405–410.
- [10] X. Yang and Y. Hu, "A scalable index architecture for supporting multi-dimensional range queries in peer-to-peer networks," in *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2006.
- [11] F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, "Connectivity based k-hop clustering in wireless network," *Telecommunication Systems*, pp. 205–220, 2003.
- [12] L. D. Penso and V. C. Barbosa, "A distributed algorithm to find k-dominating sets," *Discrete Applied Mathematics*, vol. 141, pp. 243–253, 2004.
- [13] S. Yang, J. Wu, and J. Cao, "Connected k-hop clustering in ad hoc networks," in *Proceedings of the 2005 International Conference on Parallel Processing (ICPP'05)*, 2005.
- [14] Y. Wu and Y. Li, "Construction algorithms for k-connected m-dominating sets in wireless sensor networks," in *9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2008)*, Hong Kong, China, 2008.