

Zone Sharing: A Hot-Spots Decomposition Scheme for Data-Centric Storage in Sensor Networks

Mohamed Aly, Nicholas Morsillo, Panos K. Chrysanthis, and Kirk Pruhs
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260, USA
{maly, nwm1, panos, kirk}@cs.pitt.edu

ABSTRACT

In the resource over-constrained environment of sensor networks, techniques for storing data locally in sensor nodes have been proposed to support efficient processing of ad-hoc queries. These data-centric storage (DCS) techniques differ in the method of mapping events to sensors, but all of them fail to deal with storage hot-spots due to irregular sensors deployments, and(or) data distributions. In this paper, we present Zone Sharing (ZS), a novel distributed scheme for the decomposition of storage hot-spots. We apply ZS to the DIM scheme, which has been shown to be the best among all DCS schemes. ZS locally detects hot-spots and tries to evenly distribute their loads among the sensor nodes in the network. Simulations have shown the efficiency of ZS in decomposing small to moderate sized hot-spots without imposing an additional energy load to the network nodes.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*Distributed Databases*

General Terms

Algorithms, Design, Performance

Keywords

Sensor Networks, sensor databases, data-centric storage, storage hot-spots

1. INTRODUCTION

In a typical sensor network, sensors are responsible of sensing one or more phenomenon. The reading of one or more sensors composes an *event*. Thus, an event type corresponds to a set of attributes, one for each phenomenon under concern. Applications monitor these events by submitting queries. One way to handle such queries is to propagate events to base stations, mostly outside of the network,

where queries are submitted and executed. However, this approach may be more appropriate to answer *continuous queries* which mostly process all generated events over a period of time. For *ad-hoc queries*, this approach is extremely inefficient because ad-hoc queries typically require a small set of the generated events at some point in time. Propagating all events to the base stations would be a major waste of energy that impacts the lifetime of sensor nodes. Further, more energy is wasted if ad-hoc queries can be submitted to any node in the network as such queries will need to be propagated all the way to the base stations for execution and their results have to be sent back to the querying nodes.

In order to improve the lifetime of sensor nodes and consequently the *quality of data (QoD)* of ad-hoc queries, the idea of *data-centric storage (DCS)* has been proposed for maintaining the events within the sensor network rather than storing them on base stations [5]. DCS techniques define an owner for each event, which is the sensor that will be responsible of storing the event. This event-to-sensor mapping is based on the attributes values of an event. For example, the mapping is done using hash tables in DHT [5] and GHT [2], or using k-d trees in DIM [1].

Irregularity in terms of sensors deployment or data distribution represents a vital issue in DCS techniques [6]. The problem arises when the geographic node distribution does not match the storage load distribution. In this case, a high percentage of the load is assigned to a relatively small portion of the sensor nodes. We call this problem a *storage hot-spot*. Such hot-spots represent a storage, as well as energy, bottleneck in the sensor network. The presence of hot-spots leads to increasing the dropping rate of events by sensors falling in the hot-spot area. Consequently, this decreases the *QoD* of the different queries targeting the hot-spot events. Also, insertions and queries aiming the hot-spot area are propagated through a relatively small number of routes (those leading to such area). This results in the quick depletion of the energy of sensors falling on these routes (i.e. their death), thus, leading to *network partitioning* and consequently reducing the *network lifetime*. Lost stored events and reduced coverage due to failed sensor nodes additionally decreases the *QoD*.

Current DCS schemes fail to effectively cope with the hot-spots problem, although some possible solutions for *irregular sensors deployments* were suggested, such as routing based on virtual coordinates, or using heuristics to locally adapt to irregular sensor densities [6]. In this paper, we propose a solution to the hot-spot problem due to *irregular data distribution* in DCS schemes. We will present our solution in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DMSN'05, August 29, 2005, Trondheim, Norway.

Copyright 2005 ACM 1-59593-206-2/05/0008 ...\$5.00.

the context of the DIM scheme, which has been shown to exhibit the best performance among the DCS techniques. In DIM, a sensor node is defined as a leaf of a k-d tree [4] and is assigned a binary address (*zone*) based on its geographic location. The address size (in bits) of each sensor is equal to its depth (level number) in the tree. Events are mapped to binary codes based on their attributes values and they are routed to their owners using the Greedy Perimeter Stateless Routing (GPSR) algorithm [3]. Due to its high performance compared to other DCS schemes, many researches have recently focused on improving the different aspects of DIM, for example the localization scheme presented in [7].

The intuitive idea of our proposed solution of *Zone Sharing (ZS)* is that a node experiencing high load compared to its neighbors is a good indication of a storage hot-spot and it is reasonable to share its load with one of its less-loaded neighbors. In the specific terms of DIM, a high-loaded sensor node attempts to split its owned zone with one of its less-loaded neighbors. Based on the maximum number of times a zone could be shared, we present two ZS flavors: *Single-Hop Zone Sharing* and *Multiple-Hop Zone Sharing*.

The paper experimentally shows that the main advantages of ZS are:

- Increasing the *QoD* by distributing the hot-spot events among a larger number of sensors.
- Increasing the *energy savings* by balancing energy consumption among sensor nodes.
- Increasing the *network lifetime* for networks of small and medium sizes by reducing the average number of dead nodes.

The paper is organized as follows: The next section describes and analyzes the ZS scheme. Experimental results are provided in Section 3 and Section 4 concludes the paper.

2. ZONE SHARING

In this section, we describe two ZS schemes. We first illustrate the basic ZS idea using a simple example.

2.1 Basic Idea

Figure 1 shows a typical scenario for the zone sharing process. In the k-d tree on the LHS, N0 (address = 0) is experiencing high storage load compared to its neighbors, N1 (address = 11) and N2 (address = 10). N2 has a smaller storage load than N1. The difference of load between the two subtrees is considered as a potential hot-spot indication in the left subtree. Therefore, in order to cope with this hot-spot that is about to be formed in N0, node N2 passes the responsibility of its original zone to N1 and takes responsibility of a portion (around half) of N0's zone. The k-d tree takes the form presented on the RHS of Figure 1.

Note that the move of N2 is only *logical*, i.e., N2 still keeps in its original geographic location (assuming nodes are stable), but, it takes responsibility of another zone whose code value is different from its binary address value. Also, it is assumed that each node has enough energy to send and receive events during this logical move.

The above procedure is used to decompose storage hot-spots as follows: In case a hot-spot arises in a set of sensor nodes, the *border nodes*, i.e., those nodes falling on the border of the hot-spot, will trade pass responsibility of portions

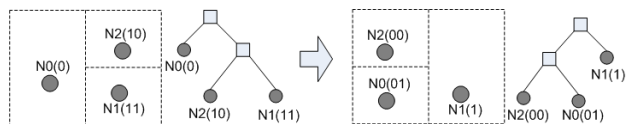


Figure 1: Zone sharing simple example

of their zones to some of their less-loaded neighbors, not falling in the hot-spot area, as described above. This will lead to decreasing the size of the hot-spot by removing these border nodes outside from the hot-spot. Then, these nodes that were previously on the hot-spot border will now act as the less-loaded neighbors for the new border nodes, and subsequently, receive part of the load of these border nodes. This process continues till it totally decomposes the hot-spot and distributes its load on a larger number of sensor nodes.

But many questions need to be answered to efficiently implement zone sharing, such as: When can a node tell that it is falling in a hot-spot? How does a node know the neighbors information? With which neighbor to trade the excess capacity? How much load to trade? How many times a given zone can be traded? What changes need to be made to GPSR to account for the zone migration? In the following subsections, we answer these questions in turn.

2.2 Distributed Migration Criterion (DMC)

A node experiencing an unusual load can split its owned zone with one of its neighbors. In such case, the node's depth in the k-d tree is increased by 1 (i.e., its address size is increased by 1 bit). A neighbor of this node is logically moved to share its zone. As this neighbor was another leaf in the tree, it passes its original zone to one of its siblings, and the tree depth of this last decreases by 1 bit.

The above process can be described as a migration of a given node from its original level to a lower one in the k-d tree. Thus, we call the original node that splits its zone **the donor** (N0 in Figure 1) as it donates almost half of its load, while its neighbor that takes part of that zone is called **the migrator** (N2 in Figure 1) as it migrates from a zone to another. The node that receives the original migrator's zone is called **the receiver** (N1 in Figure 1).

We define l_x to be the storage load of node x, while e_x is meant to be the energy level of node x. Therefore, in Figure 1, the original load of the migrator, which is passed to the receiver, is referred to as $l_{migrator}$, while the total load of the donor, before the migration process, is l_{donor} . The amount of load that the donor passes to the migrator after the migration of this last is defined as the *traded zone*, T . It is expressed in terms of the number of traded messages (Note that an event represents one message). The value of T is solely defined by the donor, based on the distribution of its storage load, in a way that balances the storage between itself and the migrator after the migration process. In case the load is uniformly distributed in the hot-spot zone, T can be considered as half of the original load of the donor, l_{donor} .

In order to define which neighbor to split zone with, we need to relate the loads, as well as the energy levels, of the three nodes involved in the migration process in a reasonable way that maximizes the profit gained from migration. Therefore, we provide a set of inequalities to be *locally* applied by the three nodes. These equations represent neces-

sary conditions that must be fulfilled to justify zone sharing. In these equations, an energy unit represents the amount needed to send one message. The fraction r_e is the amount of energy consumed in receiving one message (it is always less than one energy unit):

$$\frac{l_{donor}}{l_{migrator} + l_{receiver}} \geq C_1 \quad (1)$$

$$\frac{T}{l_{migrator}} \geq C_2 \quad (2)$$

$$\frac{T}{e_{donor}} \leq E_1 \quad (3)$$

$$\frac{l_{migrator} + r_e * T}{e_{migrator}} \leq E_2 \quad (4)$$

$$\frac{l_{migrator} * r_e}{e_{receiver}} \leq E_3 \quad (5)$$

where C_i 's and E_j 's are constants representing storage ratio and energy ratio thresholds, respectively.

The first two equations are concerned with relating the storage loads of the three nodes. Equation (1) states that the pre-migration load of the donor should be much bigger than the post-migration load of the receiver. Such constraint is needed in order to guarantee that no migration oscillation would occur. It should be applied by both the donor and the receiver. The value of the constant C_1 should be greater than or equal to 2 in order to make sure that the donor is really falling within a storage hot-spot. Equation (2) states that the post-migration load of the migrator is bigger than its pre-migration load. This is needed in order to gain some profit from the whole migration process. This equation should be applied by the migrator. It is obvious that C_2 should take values greater than or equal to 2 in order to avoid cyclic migrations.

Equations (3) to (5) relate the energy levels of the three nodes, before and after the migration process. Equation (3) states that the energy consumed in transferring the traded zone is much less than the donor's energy level prior to the migration process. It is applied by the donor. Equation (4) states that the energy consumed in sending the migrated zone, as well as receiving the shared zone, is less than the total energy amount owned by the migrator pre-migration. It is only applied by the migrator. The last equation, Equation (5), states that the energy consumed by the receiver in receiving the original load of the migrator is less than its available energy before migration. It is obvious that this last is applied by the receiver. These inequalities are needed to make sure that the energy amount consumed in the migration process will not cause the death, or the approach of death, of one or more of the nodes involved in the migration process. The values of the E_j thresholds should be less than 0.5.

To be able to localize the evaluation of the above equations, each node is supposed to maintain load information, in terms of in terms of storage and energy, of its neighbors. Thus, in case of falling in a storage hot-spot, the node will be able to select the best candidate neighbor to split its owned zone with. The periodic messages exchanged between neighbors to maintain the DIM structure, as well as insertion and query messages, can be piggybacked with such information. Each node periodically checks its storage level and applies the migration criterion in case this level exceeds a given threshold.

The migration process, as described above, needs some distributed decision making among sensors involved in migration. For such purpose, a *three-way hand-shaking* procedure must be applied in case a node decides to share its zone with one of its neighbors. First, the donor should decide which neighbor to be the migrator and contact this last with a *Request to Migrate* message (RTM). In case the candidate migrator is able to migrate, it should select an appropriate receiver and inform it about its migration decision. In case the receiver accepts this decision, the migrator replies to the donor with an *Accept to Migrate* message (ATM). Note that the RTM and ATM could be piggybacked on other messages, or sent explicitly. The overhead of such messages is negligible compared to that of the actual migration process.

2.3 Single-Hop Zone Sharing (SHZS)

Now that we defined the DMC, we need to decide how will the hot-spot decomposition process take place. This can be done by determining the number of times a single zone can be traded. In order to minimize the changes made to the tree, we limit the maximum depth change of any zone to 1. In other words, each zone can be shared only *once*. A receiver cannot share the original migrator's zone another time. Hence, we call this ZS version *Single-Hop Zone Sharing*.

As every zone will be at most one hop further from its original location, the original GPSR can be used to query the different zones and the original donor node will just forward the query, or the insertion, to the migrator in case the queried zone has been already donated. This will enable us to use the same DIM scheme with minimal changes made for the hot-spot decomposition purpose.

2.4 Multiple-Hop Zone Sharing (MHZS)

Although the SHZS algorithm is quite simple, it has three drawbacks when dealing with large hot-spots. First, with increasing storage hot-spots sizes, the neighbors of the overwhelmed node will most probably be falling in, or close to, the hot-spot, thus suffering from the same symptom. Therefore, the DMC will be hard to be satisfied by such nodes. As the hot-spot size increases, border nodes will be extremely loaded as well as their neighbors, thus complicating the DMC satisfaction. Further, when a border node passes a portion of its load to a neighbor, it will be still falling in the hot-spot, thus, unable to receive any load from other nodes closer to the center of the hot-spot. Hence, the DMC will not lead to the hot-spot decomposition.

Second, the fact that a shared zone cannot be shared again complicates the scheme in the case of dealing with large hot-spots. At some point, all the neighbors of the hot-spot node will be responsible of zones that were already falling under this node's responsibility. Then, this last receives further events by dropping old ones, leading to decreasing the QoD.

A third complication is regarding the GPSR algorithm in the case of events insertion. Assuming that a donor selects to share the zone that has more upcoming events to be inserted in the future, all such insertions will first pass by the donor before going to the migrator. This will impose a large energy consumption burden on the donor node.

From the above points, it is obvious that the SHZS algorithm is best suited for small hot-spots. In order to extend the zone sharing idea to handle larger hot-spots, we need to relax the single hop sharing assumption. Thus, we intro-

duce another ZS version where a zone can be shared more than once. This implies that a given zone can encounter a tree depth change of any size. We call this version of zone sharing *Multiple-Hop Zone Sharing*.

As a zone can be moved several hops away from its original node, keeping GPSR with no changes will involve the original donor, as well as subsequent ones, in all insertions and queries concerning the shared zone. This would be an extreme overhead and it would be proportional to the overall depth change of the shared zone. Hence, GPSR must be augmented by some means to figure out that a zone has been shared and moved away from its original location.

For such purpose, we assume that each node maintains a *Shared Zones List* containing three entries: zone address, original donor, and final migrator. Upon zone sharing, the donor sends the shared zone address, its name and the migrator's name to all its neighbors. Thus, each node will be aware of zones traded by its neighbors.

In case of multiple sharing of the same zone, the old migrator becomes the new donor. It sends the zone address, the original donor, as well as the new migrator to all of its neighbors. Thus, it has to check its shared zones list first and send the entry corresponding to the zone under concern after updating the final migrator entry with the new migrator.

In case a node receives a shared zone entry that is already present in its shared zones list, it updates its list with the new entry. This means that a given zone has been re-shared. This update guarantees that a shared zone will have similar entries in all shared lists containing information about this shared zone. The node then forward the shared zone entry to its neighbors.

In routing an event, GPSR first checks the shared zones list with the zone address. In case an entry for such zone is found, this means that the original destination of the zone has been changed to a new one. Therefore, the destination node found in the shared zones list entry of the zone is used to explicitly update the destination field of the message to be routed to its new value. GPSR then uses the new destination address to forward the message using the best path to the final migrator. A flag is updated in the message to indicate that its original destination has been already changed to its current one to avoid further overhead lookup in subsequent nodes.

As we are constraining each node to send shared zones information only to its neighbors, the size of a shared zone entry will be relatively small. It is easy to prove that an n -times shared zone will be at most present in the shared zone lists of nodes of $\theta(n)$ hops away from its final destination. Thus, the overhead that GPSR will take to search for a shared zone in its list is relatively small. Also, for shared zones, shared lists search occurs only once in the furthest node, containing shared zone information, from the final destination. As mentioned above, upon seeing that the destination flag set in the message, GPSR in the following nodes uses such destination immediately without searching the list.

From the above, ZS tries to decompose the hot-spot storage load across the network. The decomposition process takes place from the hot-spot borders and going all the way to the hot-spot center. The following section presents simulation results that show ZS ability of decomposing storage hot-spots of different sizes.

3. SIMULATION RESULTS

In order to measure the ZS performance, we created a simulator for a typical sensor network applying DIM, as presented in [1]. We also simulated GPSR to be used as the routing protocol. Then, we added the ZS functionality to such network to compare the effect of applying ZS versus using pure DIM.

In our simulation, we tried to use the same experimental setting used in [1] to get similar DIM performance. Thus, we simulated networks of sizes ranging from 50 to 300 sensors, each having an initial energy of 50 units, a radio range of 40m, and a storage capacity of 15 units. The sensors locations were drawn from a uniform distribution. The service area was computed such that each node has on average 20 nodes within its nominal radio range.

We model hot-spots by a using a uniform distribution to represent sensors locations, while using a skewed distribution of events among the attributes ranges. We assume that each sensor has a limited storage capacity and that it replaces the oldest event in its memory in case it is already full.

Concerning the DMC parameters, we chosen a value of 2 for the C_1 and C_2 constants of the first two equations (storage load equations). For the energy load equations, we set the E_1 , E_2 and E_3 constants to 0.3. Note that these are just typical values for such thresholds. For an exact performance of ZS, extensive binary search among the different combinations of such constants should be made. Concerning the r_e value, we assumed that message receiving takes half an energy unit (One unit = amount needed to send an event). We assumed a network with a single hot-spot, where a percentage varying from 10% to 50% of the events values fell into within 10% of the attributes ranges.

The simulation consisted of two phases: *the insertion phase* and *the query phase*. During the insertion phase, each sensor initiates 5 events, according to the predefined hot-spot size, and forward each event to its owner. In the query phase, each sensor generates queries of sizes ranging from 10% to 100% of the attributes ranges.

The results of the simulations are shown in the following figures. In these figures, we compare the performance of the pure DIM versus that of SHZS, as well as MSHZS, with respect to five performance measures. Note that we only show some of our findings due to space constraints.

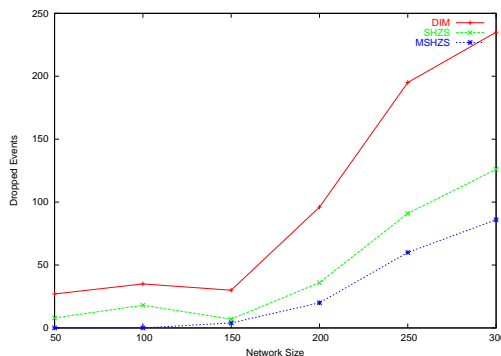


Figure 2: Number of dropped events for networks with a 30% hot-spot after insertion phase

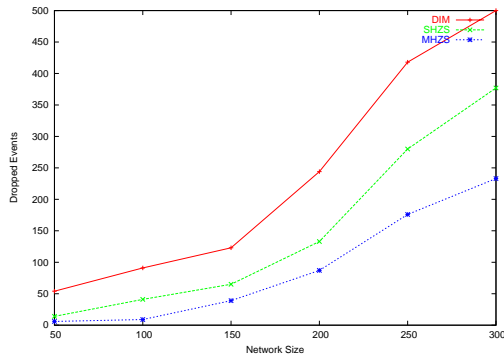


Figure 3: Number of dropped events for networks with a 50% hot-spot after insertion phase

R1. Data Persistence: Figures 2 and 3 present the total number events dropped by all network nodes after the insertion phase in networks with hot-spots of sizes of 30% and 50%, respectively. For the three simulated schemes, the number of dropped events is quite low and almost constant for networks of small sizes (less than 150 nodes), while it increases linearly for larger network sizes. However, the ZS schemes, especially MHZS, highly improve the performance by decreasing the number of dropped events for all network sizes.

The above result proves that ZS improves the average lifetime of an event to be stored in the network, which is the time that an event passes in the network before being dropped. Therefore, for queries aiming an arbitrary set of events, the network will be able to efficiently provide answers to such queries for longer time periods.

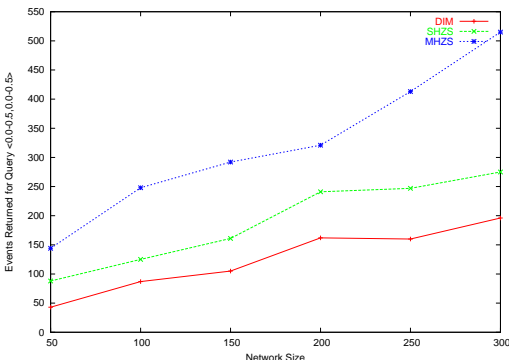


Figure 4: Query size of a 50% query for networks with a 50% hot-spot

R2. Quality of Data: Figures 4 and 5 show the average query sizes of 50% and 70% of the attribute ranges, respectively, for networks with 50% and 40% hot-spots, respectively. It is clear that ZS schemes, especially MHZS, improve the QoD by dropping less information, as described in R1, thus increasing the number of events resulting in each query. Note that the gap between pure DIM and ZS schemes, in terms of resulting query sizes, increases with the increase of the hot-spot size.

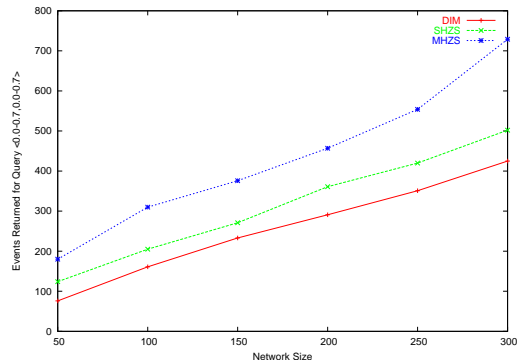


Figure 5: Query size of a 70% query for networks with a 40% hot-spot

This result has a very important implication on the *data consistency* of the sensed data output from a network experiencing a hot-spot. The success of ZS schemes in decomposing the hot-spots results in improving the network ability to keep a higher portion of the hot-spot data. This ameliorates the degree of correctness of any aggregate functions made on the network readings, for example, an average of the temperature or pressure values where a high percentage of the data is falling within a small range of the total attributes range. We consider this to be a good achievement compared to the pure DIM structure.

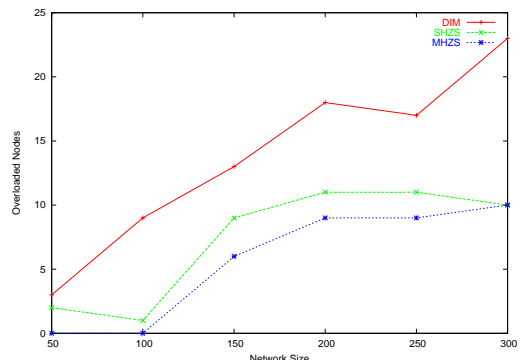


Figure 6: Number of overloaded nodes for networks with a 40% hot-spot after insertion phase

R3. Load Balancing: Figure 6 presents the number of overloaded nodes after the insertion process for networks with a hot-spot of size of 40%. By overloaded nodes, we mean the nodes having full memories. Thus, in our simulation, this means a node having 15 events in its cache. The figure shows that the performance of networks applying the ZS schemes is much better than those applying the pure DIM scheme. The MHZS achieves a ratio of improvement ranging from 25% to 50% based on the network size.

The previous results prove that ZS schemes, especially MHZS, improve the network ability to maintain a higher portion of the hot-spot events by achieving a better balancing of such events among the network nodes.

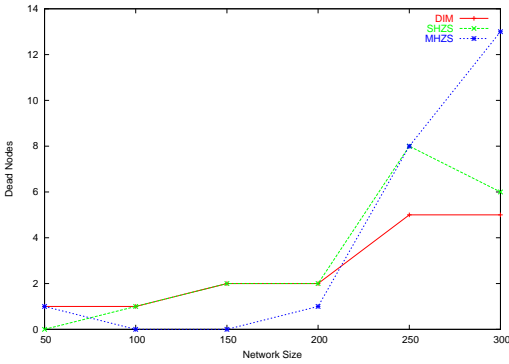


Figure 7: Number of dead nodes for networks with a 50% hot-spot after insertion phase

R4. Network Lifetime: Figure 7 presents the number of dead nodes after the insertion process for networks with hot-spot size of 50%. The number is negligible for the three schemes in case of small to medium networks (less than 200 node). Then, this number increases in case of larger networks. The performance of ZS schemes is slightly better than the pure DIM in small and medium networks, while it is lower than DIM for larger networks in this experiment (although both of them achieve a small number of dead nodes compared to the whole network size). This result shows that the additional work done by the sensors in the ZS process increases the network lifetime for small to medium networks, in addition to the increase in the QoD in all cases shown above. Recall that in this experiment, MHZS was unrestricted in terms of decomposing hot-spots over large number of hops. Hence, the performance could have been improved for large networks by defining an upper bound of the number of times (hops) a zone could be traded or by dynamically adjusting the E_j values of the DMC. We plan to experiment with these extensions in our future work.

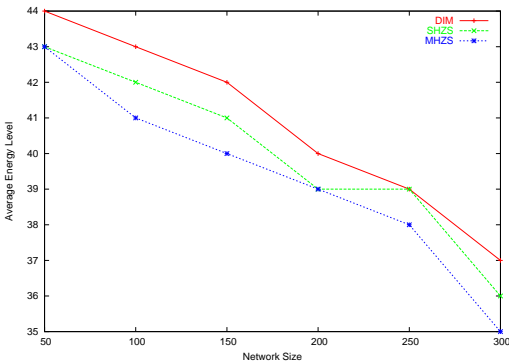


Figure 8: Average node energy level for networks with a 50% hot-spot after insertion phase

R5. Energy Consumption: Figure 8 presents the average node energy level after the insertion process for networks with hot-spot size of 50%. The figure shows that this average energy level decreases with the increase of the network size. Introducing ZS schemes slightly decreases the average sensor energy by a small amount compared to the correspond-

ing average in the pure DIM case. However, ZS increases the number of sensors taking responsibility of the hot-spot events. Hence, ZS balances the energy consumption among a larger number of sensors in the network compared to fewer sensor nodes in the pure DIM. This can be viewed as decreasing the variance of the nodes energy levels compared to the pure DIM.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we presented Zone Sharing (ZS), a novel scheme for decomposing storage load of hot-spots in Data-Centric Storage sensor networks. ZS is based on locally detecting the formation of a hot-spot, and iteratively trying to break off the hot-spot load, starting from the hot-spot nodes toward their neighbors. We described two types of ZS schemes, based on the level of sharing of the different zones: *Single-Hop* and *Multiple-Hop*.

We have shown experimentally that applying ZS schemes to the DIM structure achieves good performance in the case of skewed data distributions. ZS is able to decompose small to moderate sized hot-spots without adding an extra additional load on the different sensor nodes. This leads to having a better usage of the sensed data, i.e. improve the QoD. The last result implies an amelioration of the profit gained from the sensor network throughout the network lifetime.

In the future, we would like to develop a new global scheme for incremental load balancing throughout the network lifetime. Such scheme should act as a storage hot-spots avoidance mechanism, instead of a storage hot-spots detection and decomposition mechanism like ZS.

Acknowledgment

We thank the reviewers for their helpful comments. This work has been supported by NSF grants CCR-0098752, ANI-0123705, ANI-0325353, CCF-0448196, and CCF-0514058.

5. REFERENCES

- [1] Xin Li, Young Jin Kim, Ramesh Govidan, and Wei Hong, "Multi-dimensional Range Queries in Sensor Networks". In *Proc. of the ACM SenSys*, 2003.
- [2] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govidan, and Scott Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage". In *Proc. of the 1st ACM Intl. Workshop on Wireless Sensor Networks and Applications*, 2002.
- [3] Brad Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Sensor Networks". In *Proc. of the ACM Mobicom*, 2000.
- [4] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching". In *CACM* 18(9), 475-484, 1975.
- [5] Scott Shenker, Sylvia Ratnasamy, Brad Karp, Ramesh Govidan, and Deborah Estrin, "Data-Centric Storage in Sensornets". In *Proc. of the 1st Workshop on Hot Topics in Networks*, 2002.
- [6] Deepak Ganesan, Sylvia Ratnasamy, Hanbiao Wang, and Deborah Estrin, "Coping with Irregular Spatio-temporal sampling in Sensor Networks". In *Proc. of the 2nd Workshop on Hot Topics in Networks*, 2003.
- [7] Lin Xiao and Aris Ouksel, "Tolerance of Localization Imprecision in Efficiently Managing Mobile Sensor Databases". In *Proc. of the 4th ACM MobiDE*, 2005.