Decomposing Data-Centric Storage Query Hot-Spots in Sensor Networks †

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

Abstract

Arising when a large percentage of queries is access-ing data stored in few sensor nodes, query hot-spots reduce the Quality of Data (QoD) and the lifetime of the sensor network. All current In-Network Data-Centric Storage (IN-DCS) schemes fail to deal with query hot-spots resulting from skewed query loads as well as skewed sensor deploy-ments. In this paper, we present two algorithms to locally detect and decompose query hot-spots, namely Zone Parti-tioning (ZP) and Zone Partial Replication (ZPR). We build both algorithms on top of the DIM scheme, which has been shown to exhibit the best performance among all INDCS schemes. Experimental evaluation illustrates the efficiency of ZP/ZPR in decomposing query hot-spots while increasing QoD as well as energy savings by balancing energy con-sumption among sensor nodes.

1   Introduction

Sensor networks provide us with the means of effectively monitoring and interacting with the physical world. As an example of the type of sensor network application that concerns us here, consider an emergency/disaster scenario where sensors are deployed in the area of the disaster [18]. It is the responsibility of the sensor network to sense and store events of potential interest. An *event* is composed of one or more attributes (e.g., temperature, carbon monoxide level, etc.), the identity of the sensor that sensed the event, and the time when the event was sensed. As first respon-ders move through the disaster area with hand-held devices, they issue queries about recent events in the network. For example, the first responder might ask for the location of all sensor nodes that recorded high carbon monoxide levels in the last 15 minutes, or he might ask whether any sen-sor node detected movement in the last minute. Queries

are picked up by sensors in the region of the first respon-der. The sensor network is then responsible for answering these queries. The first responders use these query answers to make decisions on how to manage the emergency.

The *ad-hoc queries* of the first responders will generally be *multi-dimensional range queries* [10], that is, the queries concern sensor readings that were sensed over a small time window in the near past and that fall in a given range of the attribute values. In-Network Storage (INS) is a storage technique that has been specifically presented to efficiently process this type of queries. INS involves storing events locally in the sensor nodes. Storage may be in-network because it is more efficient than shipping all the data (i.e., raw sensor readings), as well as the queries, out of the net-work (for example to base stations), or simply because no out-of-network (or in-network) storage server is available (for example, in our application, it can be hard to imme-diately provide powerful base stations in the disaster area). As the query load is composed of range queries, *local stor-age*, where each sensor stores the events it generates, is not of interest as querying such type of storage schemes will re-quire flooding every query to all sensor nodes, which incurs a very high energy consumption overhead.

All INS schemes already presented in literature were *Data-Centric Storage (DCS)* schemes [16], thus, based on a function from events to sensors that maps each event to an owner sensor based on the value of the attributes of that event. The owner sensor will be responsible for storing this event. The owner may be different than the sensor that orig-inally generated the event. Although many INDCS schemes have been presented to date like DHT [16] and GHT [14], DIM has been shown to exhibit the best performance among all proposed INDCS schemes in dealing with sensor net-works whose query loads are basically composed of ad-hoc queries [10].

In the DIM [10] scheme, the events-to-sensors mapping is based on a k-d tree [4], where the leaves $\mathcal{R}$ form a parti-tion of the coverage area, and each element of $\mathcal{R}$ contains either zero or one sensor. The formation of the k-d tree

consists of rounds. Initially, $\mathcal{R}$ is a one element set containing the whole coverage area. In each odd/even round r, each region $R \in \mathcal{R}$ that contains more than one sensor is bisected horizontally/vertically. Each time that a region is split, all the sensors in that region have a bit appended to that address specifying which side of the split that the sensor was on. Thus, the length of a sensor's address (bit-code) is its depth in the underlying k-d tree. When a sensor generates an event, it maps such event to a binary code based on a repetitive fixed uniform splitting of the attributes ranges in a round robin fashion. For our purposes, it is sufficient for now to consider the cases where the event consists of only one attribute, say temperature. Then, the high order bits of the temperature are used to determine a root-to-leaf path in the k-d tree, and if there is a sensor in the region of the leaf, then this sensor is the owner of this event. Due to the regularity of regions in this k-d tree, the routing of an event from the generating sensor to the owner sensor is particularly easy using the Greedy Perimeter Stateless Routing (GPSR) algorithm [8].

A major problem in DIM is that of *query hot-spots*. Query hot-spots may occur in DIM if the sensors are not uniformly distributed. Query hot-spots may also occur in case of a skewed query workload. In both cases, relatively many queries are requiring data stored in a relatively small number of the sensors, thus, a query hot-spot is formed in the geographic area of these sensors. For example, if there was only one sensor on one side of the first bisection, then half of the query load would be accessing this sensor (assuming a uniformly distributed query workload). The presence of a query hot-spot leads to increasing the energy consumption rate of overloaded sensors. More critically, the sensors in and near the hot-spot may quickly run out of energy, due to the high query load imposed to them (in addition to event insertions). This results in a loss of the events generated at these sensors, the events stored at these sensors, and possibly a decrease in network connectivity. Increased death of sensors results in decreasing the coverage area and causes the formation of coverage gaps within such area. This consequently decreases the QoD. Additionally, Queries for events in a query hot-spot may be delayed due to contention (among themselves, as well as with event insertions) at the hot-spot sensors and the surrounding network of sensors. Certainly, it is not desirable to have a storage scheme whose performance and QoD guarantees rest on assuming a uniform distribution of both, sensor locations and query load. To our knowledge, no previous solutions have been provided in literature to cope with the query hot-spots problem.

In this paper, we propose two algorithms locally solving the query hot-spots problem in the DIM framework: *Zone Partitioning (ZP) and Zone Partial Replication (ZPR)*. ZP considers a node, having a frequently accessed zone
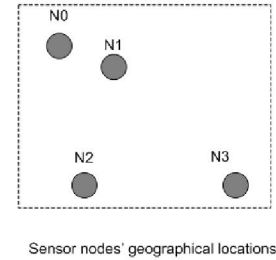

Sensor nodes' geographical locations

Figure 1. Initial network configuration

compared to its neighbors' zones, a good indication of a query hot-spot. A reasonable solution for such case is to force such node to split its owned zone with one of its less-accessed neighbors. For the case where the access frequency is not homogeneous among the subranges of the frequently accessed zone, we present a second algorithm, ZPR, to replicate the events of the highly accessed subranges in a larger number of sensor nodes in order to reduce the total number of queries accessing the query hot-spot location.

Experimental evaluation shows that the main advantages of applying ZP/ZPR on top of DIM are:

- Increasing the QoD by distributing the events stored at sensors falling in the query-hotspot among a larger number of sensors, thus, increasing the data persistence as well as its availability in the network.

- Increasing the *energy savings* by balancing energy consumption among sensors in case of a query hot-spot.

The paper is organized as follows. Section 2 provides an overview on DIM. Section 3 and 4 describe the proposed ZP and ZPR algorithms, respectively. Experimental results are discussed in Section 5, while related work is presented in Section 6. Finally, Section 7 concludes the paper.

2 Overview on DIM

In this section, we will briefly describe the components of DIM using a simple example. We assume that the sensors are arbitrarily deployed in the convex bounded region R. We assume also that each sensor is able to determine its geographic location (i.e., its x and y coordinates), as well as, the boundaries of the service area R. Each node is assumed to have a unique *NodeID*, like a MAC address. Sensor nodes are assumed to have the capacity for wireless communication, *basic* processing and storage, and they are associated with the standard energy limitations.

The main components of any DCS scheme are: the *sensor-to-address* mapping that gives a logical address to each sensor, and the *event-to-owner-sensor* mapping that determines which sensor will store the event. DIM implements these two mappings statically. The sensor-to-address
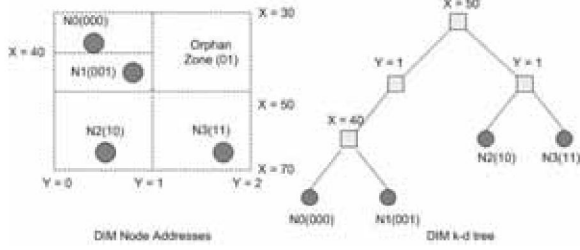
Figure 2. DIM k-d tree

Figure 3. ZP example

mapping is done at the start of the network operation based on the geographic locations of sensors. Figure 2 shows the k-d tree that the DIM forms for the simple network given in Figure 2. In Figure 2, the orphan zone (01) is assumed to be delegated to node 001, which is the least loaded among its neighbors. Periodic messages are exchanged between sensor nodes to maintain the DIM k-d tree structure.

We now move on to the event-to-owner-sensor mapping. The generation of the event bit-code proceeds in rounds. As we proceed, there is a range R_j associated with each attribute j of the event. Initially, the range R_j is the full range of possible values for attribute j. We now describe how a round $i \geq 0$ works. Round i, determines the $(i+1)^{st}$ high order bit in the code. Round i depends on attribute $j = i \bmod k$ of the event, where k is the number of attributes in the event. Assume that the current value of R_j is $[a, c]$, and let $b = (a+c)/2$ be the midpoint of the range R_j. If the value of attribute j is in the lower half of the range R_j, that is in $[a, b]$, then the ith bit is 0, and R_j is set to be the lower half of R_j. If the value of attribute j is in the upper half of the range R_j, that is in $[b, c]$, then the ith bit is 1, and R_j is set to be the upper half of R_j.

Now that we have described our underlying environment, we start presenting the query hot-spots decomposition algorithms that we propose in this paper.

3 Zone Partitioning (ZP)

In this section, we describe ZP, which is the first algorithm we present for decomposing query hot-spots in DIM. As we have discussed in previous sections, a query hot-spot represents a skewness of the query load toward a given path of the k-d tree. Thus, ZP is based on continuously checking for such skewness and trying to rebalance it when it is formed at any subset of sensors. We first illustrate the basic ZP idea using a simple example.

3.1 Illustrative Example

Figure 3 shows a typical scenario explaining how ZP works. In the Figure, a circle represents a sensor node and an arrow represents a hop on a query path. A black node is one that is storing data accessed by queries, while a white
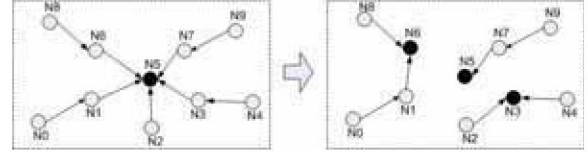
node is accessed by no queries in the query load. In the sensor network on the left hand side, queries sent from sensor nodes N0, N2, N4, N8, and N9 require data falling in the storage range responsibility of N5. After knowing that none of its neighbors is accessed by queries, N5 determines that it is responsible for a hot range of attribute values, thus, falling in a query hot-spot. Subsequently, node N5 partitions the responsibility of its original zone between itself, node N3, and node N6. After the zone partitioning, queries take the paths described in the right hand side of the Figure. Note that this zone partitioning is only logical, hence, nodes N3 and N6 keep their original zones (i.e., addresses) and each of them takes responsibility of another zone whose code value is different from its binary address value. Note that Nodes N3 and N6 must have enough memory space to store the newly received events belonging to the hot partition passed by N5.

In the above ZP process, we call the node that passes (donates) responsibility of part of its hot zone **the donor** (N5 in Figure 3). The neighbors that take parts of the partitioned zone are called **the receivers** (N3 and N6 in Figure 3).

Now that we have presented the basic ZP idea, we show the actual ZP components in the following subsections.

3.2 Local Detection of Query Hot-Spots

In order to locally detect hot-spots, each node keeps track of the frequency of accesses of its events. The tuple representing every stored event is appended by a counter called the $access\_freqeuency$. This counter represents the number of queries accessing such event over a given time period (window), w. The node resets all counters at the start of w and increments the counter associated with an event every time it receives a query requesting this event during w.

Each sensor node continuously computes the Average Access Frequency, $AAF(Z_k)$ of each of its zones Z_k, which is the average of the access frequencies of events belonging to zone Z_k. Note that, in general, a node can be responsible for more than one zone. At this point, let us assume that events falling in each zone will be uniformly accessed. In case a node x realizes that

$$\frac{AAF(Z_i)}{AAF(Z_j)} > threshold_1 \ \forall Z_j \ stored \ in \ x$$

that is, zone Z_i is highly more accessed than all other zones, x considers this as an indication of a query hot-spot that it

experiences. Thus, x decides to split the hot zone Z_i into two partitions: Z_{i1} and Z_{i2} in a way that keeps $AAF(Z_{i1})$ and $AAF(Z_{i1})$ almost equal. Note that each of the zone addresses of Z_{i1} and Z_{i2} should be one bit longer than the zone address of Z_i, as the former zones are children of the latter in the k-d tree. Then, x keeps one of the partitions and passes the other one to a selected node of its neighbors, namely the *receiver*. We call the partition that will be passed to a receiver, the traded zone T. Note that the ratio $threshold_1$ should be larger than 1 to guarantee the real existence of a hot-spot. After electing Z_i from its zones, x compares $AAF(Z_i)$ to the $AAFs$ of its neighbors as we discuss in the next subsection.

3.3 The Partitioning Criterion (PC)

We now present the Partitioning Criterion (PC), which is a set of inequalities to be *locally* applied by the donor to select the best receiver among its neighbors. The PC inequalities relate the loads, as well as the energy levels, of the donor with those of its neighbors. In these inequalities, we express the traded zone, T, in terms of the number of traded messages (Note that an event represents one message). 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). We express the total storage capacity of a sensor node by S. By l_x, we mean the storage load of node x, while e_x is meant to be the energy level of node x:

$$T + l_{receiver} \leq S \tag{1}$$

$$\frac{T}{e_{donor}} \leq E_1 \tag{2}$$

$$\frac{T * r_e}{e_{receiver}} \leq E_2 \tag{3}$$

$$\frac{AAF(donor)}{AAF(receiver)} \geq Q_1 \tag{4}$$

where E_i's and Q_j are constants representing energy ratio and average access frequency thresholds, respectively.

Equation (1) represents a *Storage Safety Requirement*. It states that the sum of the pre-partitioning load of the receiver and the traded zone size should be less than the storage capacity of the receiver. Such constraint is needed in order to guarantee that the receiver will be able to store all the traded zone events. Equations (2) and (3) represent *Energy Safety Requirements*. Equation (2) states that the ratio of the energy consumed by the donor in sending the traded zone to the available donor energy is less than a given threshold. Equation (3) states that the ratio of the energy consumed by the receiver in receiving the traded zone to the available receiver energy is less than a given threshold. These inequalities are needed to make sure that the energy amount consumed in the partitioning process will not cause

the death, or the approach of death, of one or more of the nodes involved in the partitioning process. The values of the E_i thresholds should be less than 0.5. Finally, equation (4) represents the *Access Frequency Requirement*. It relates the average access frequencies of both, the donor and the receiver, and makes sure that the ratio of the first to the second is greater than a threshold Q_1. This is needed in order to guarantee that the donor is really falling within a query hot-spot, as well as to be able to choose the best receiver to partition the hot zone with. The value of Q_1 should be greater than 2.

To be able to apply the *PC*, the donor is supposed to know load information, in terms of in terms of storage, energy and average query frequency, of its neighbors. The periodic messages exchanged between neighbors to maintain the DIM structure, as well as insertion and query messages, can be piggybacked with such information. A sensor node experiencing a high frequency of accesses uses such neighbors' information to select the best receiver among them. This can be done by selecting the node that minimizes the left hand side of equation (3) while maximizing the left hand side of equation (4). Upon selecting a receiver, the donor sends a *Request to Partition (RTP)* to such receiver. In case the receiver accepts this request, it replies to the donor with an *Accept to Partition (ATP)*. Note that the RTP and ATP could either be piggybacked on other messages, or sent explicitly. The overhead of the load information exchange messages, as well as the hand shaking messages, is small compared to that of the traded zone passing messages as the formers are taking benefit of the structure maintaining messages that were presented in the original DIM scheme.

We now illustrate how the decomposition of large query hot-spots, arising in more than one sensor, takes place. Given the PC inequalities, nodes near the center of the hot-spot cannot find any receiver to partition load with, as all their neighbors are falling in the hot-spot. Therefore, the hot-spot decomposition starts from the *hot-spot borders*. Each of nodes on the borders of the hot-spot partitions its storage with one of its less loaded neighbors. These border nodes subsequently cease belonging to the hot-spot. A new set of nodes now fall on the hot-spot borders. These nodes start partitioning their storage with their less loaded neighbors. The process continues in an iterative fashion, decomposing the hot-spot more and more at each iteration, until nodes in the center of the hot-spot become normally loaded. This signifies the complete decomposition of the hot-spot.

3.4 GPSR Modifications

We now discuss the changes introduced to the GPSR algorithm to account for ZP. Based the above ZP algorithm, a receiver can re-apply the PC to partion a previously traded zone. The process can be applied more than once. At each

time, a smaller hot sub-zone is moved further away from its original node. In case of k subsequent partitioning times, keeping GPSR with no changes will involve the original donor in all insertions and queries concerning any of the k traded zones. This overhead would be proportional to the total number of hops a zone is traded. To reduce this overhead, we augmented GPSR to recognize that a zone has been traded and moved away from its original owner.

For such purpose, each node maintains a *Traded Zones List (TZL)* containing three entries: zone address, original donor, and final receiver. Upon ZP, the donor sends the traded zone address, its ID and the receiver's ID to all its neighbors. Thus, each node will be aware of zones traded by its neighbors. In case a receiver x repartitions a traded zone z into z_1 and z_2, x becomes the new donor of z. Therefore, x sends the new sub-zone address (assume it z_2), the original donor of z, as well as the new receiver of z_2 to all of its neighbors. Note that x gets the original donor of z from the entry of z in its TZL. Each of the neighbors, upon receiving such entry, should check its TZL for a previous entry for z, or any of its parent zones. In case of finding z's entry, the node overwrites its zone address by z_1. Then, the node adds a new entry for z_2 in its TZL. At the end, it forwards the z_2 entry to all its neighbors.

As we are constraining each node to send traded zone information only to its neighbors, it is easy to prove that an entry for a p-times traded sub-zone will be present in the TZLs of nodes falling on a path of $\theta(p)$ hops away from its final destination. Based on the definition of a query hot-spot, the number of zones that will be originally falling in the hot-spot will be very small. Thus, keeping the TZL represents a small storage overhead on all sensor nodes. Additionally, the computation overhead imposed on any node for searching its TZL for an entry is $O(\log p)$, which is relatively small.

Using the TZL concept, GPSR is changed as follows. Each message sent by GPSR is appended by a $dest\_changed$ bit flag and a $dest$, which is a node address entry. Originally, $dest\_changed$ is set by the message sender to 0. In routing a message (an event or a query), a node x first checks the $dest_changed$ flag. In case it is 1, x uses the $dest$ variable as an explicit destination for the message and uses the original GPSR to select the next hop for the message toward $dest$. Otherwise, x forms the message destination address z using the original DIM event to bit-code mapping and then checks its TZL for an entry whose zone address (or left most significant string of the zone address) is identical to z. In case z is found in x's TZL, x sets the $dest\_changed$ to 1 and $dest$ to the new destination of z found in the z's TZL entry. The original GPSR is then used to send the message to $dest$. In case no entry is found for z, GPSR uses z as the message destination.

Using the modified GPSR algorithm, TZL search for any p-times traded zone T occurs only once by a node which is $\theta(p)$ hops away from the final destination of T. As soon as such node updates the $dest\_changed$ and $dest$ fields in the message, $dest$ is explicitly used by GPSR in the subsequent nodes without searching the list. This has the effect of minimizing the overhead of both, energy consumption and computational load, imposed on nodes falling in the query hot-spot.

3.5 Coalescing Process

We now discuss how to cope with query load distribution changes. Based on ZP, the receiver continues to keep track of the accesses of the traded zones. In case any of such zones is not accessed for a complete time window, d, this is considered as an indication that the hot-spot has stopped to exist (or may have moved to another location). At such point, the receiver transfers the responsibility of the received zone back to its original owner. We call this the *coalescing process*. All neighboring nodes drop the traded zone entry from their TZLs. Further events belonging to, as well as queries asking for, that zone are directed to the original donor based on the original DIM and GPSR schemes. In such process, the receiver only sends recent events belonging to the previously received zone to the original donor (events inserted during d). Thus, the coalescing process is considered much cheaper than the partitioning process. In case d is large enough, we can guarantee that the hot-spot would not be formed again in the future and that the coalescing process is not causing a loss in the quality of data provided by the sensor network as the receiver drops the events inserted before d. Using such coalescing process, partitioning *oscillations*, where the responsibility of storing the zone keeps going back and forth between its original owner in DIM and other neighboring sensor nodes due to hot-spot changes, can be avoided.

From the above, ZP decomposes the query hot-spot by continuously partitioning its storage load with its neighboring sensors starting from the hot-spot borders and moving toward its center. However, in ZP, we assume that the access frequency is uniform among the subranges of the hot zone. How about in case it is skewed toward a narrow subrange? The following section addresses this question by presenting our second query hot-spots decomposition algorithm.

4 Zone Partial Replication (ZPR)

We now present ZPR, which is our second query hot-spots decomposition algorithm. ZPR mainly deals with the case where a sensor x falls in a query hot-spot and a large percentage of the queries accessing x target a small range of the total x's attribute range responsibility. In such case, a fairly limited number of events is accessed. ZPR decomposes the query hot-spot by replicating such events in neigh-

boring sensors. ZPR is meant to work in parallel with ZP such that the combination of ZP/ZPR can efficiently handle the decomposition task of query hot-spots of different sizes.

4.1 Additional PC Requirements

First, let us show how a node decides whether to apply ZP or ZPR. To do this, it is necessary to enhance the PC (the Partitioning Criterion already presented in Section 3.3) by additional conditions that help the node falling in a query hot-spot in determining which algorithm it should use for hot-spot decomposition. Specifically, we add two more *Access Frequency Requirement* inequalities to the PC.

$$\frac{AAF(partial\,subrange)}{AAF(total\,zone\,range)} \geq Q_2 \qquad (5)$$

$$\frac{size(partial\,subrange)}{size(total\,zone\,range)} \leq Q_3 \qquad (6)$$

Equation (5) relates the AAF of given subrange of the attribute ranges of the hot zone to the AAF of the entire hot zone range. Such equation indicates that almost all queries targeting the hot zone are basically asking for events falling within the hot subrange of attribute values. The threshold, Q_2 should take values close to 1, such as 0.7 to 0.9. On the other hand, equation (6) makes sure that the size of the hot sub-zone is fairly small compared to the total hot zone size. It is clear that the threshold, Q_3, should take values close to 0, such as 0.2 or less.

A node falling within the query hot-spot first tries to satisfy all 6 PC inequalities. In case it succeeds in this, then the node chooses to apply ZPR. In the node is only able to satisfy the first 4 PC inequalities, it proceeds in applying ZP as presented in Section 3. In case ZPR is chosen to be applied, the donor sends the hot sub-zone to all its direct neighbors. Each of these neighbors, upon receiving the hot sub-zone, inserts an entry for such sub-zone in its TZL. The entry is represented by the tuple (sub-zone code, donor, self address). In other words, the node indicates itself as the receiver of such sub-zone. Note that we selected to follow the same TZL technique as the one used in ZP in order to reduce special cases.

Based on the above, no further changes need to be imposed to the modified GPSR to route queries in ZPR. Upon routing a query asking for a given zone, the node checks its TZL first. In case an entry is found for such with the receiver address is equal to the node's self address, the event is simply looked up in the node's storage. When a query is answered by one of the receivers of the replicated hot zone, such receiver broadcasts the hot zone to all its neighbors. In other words, whenever a replicated hot zone is used to answer queries, we enforce such zone to be re-replicated one hop further from the original hot zone owner. Neighbors receiving such broadcast store the replicated hot zone
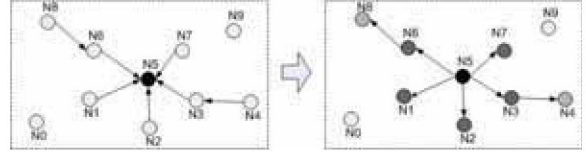

Figure 4. ZPR example

in case of space availability. Note that in case a query asks for a portion of the hot zone and the query is answered by replica node, such node only broadcasts that portion of the hot-zone. This should not happen frequently as the hot zone is small enough by definition.

Figure 4 shows a ZPR example. In the left hand side, node N5 (in black) is accessed by queries sent from nodes N1, N2, N4, N7, and N8. By applying ZPR, N5 sends the hot sub-zone events to all its direct neighbors (nodes in dark gray, i.e., nodes N1, N2, N3, N6, and N7). Thus, queries initiated by N1, N2, and N7 are answered from the initiating nodes' self storages. For queries initiated at N4 and N8, the results are first provided by N3 and N6, respectively. Each of the nodes N3 and N6, upon realizing that the replicated hot sub-zone was used in answering such queries, forward such sub-zone to N4 and N8 (filled with light gray), respectively.

4.2 ZPR Handling of Insertions

An interesting question is how ZPR handles event insertions. Recall that when any node x receives a message (in this case, an event insertion), it applies the modified GPSR presented in Section 3.4. Thus, x first checks its TZL for the destination zone (i.e., node address) of such event. In case ZPR was applied to the event's zone z, x will find an entry for z with x's address as z's final destination. In such case, x inserts the new event in its memory and proceeds with sending the event to its original owner as determined by the DIM scheme. Upon receiving a newly inserted event, the original owner re-initiates ZPR to propagate such event to all neighbors having copies of z.

It is worth mentioning that ZPR may encounter some inconsistency in the answer of some simultaneously posed queries. Consider in Figure 4, a new hot-spot event generated by node N7. At the same time, two queries are generated by N4 and N8. By the time the new event arrives at N5, N8's query would have been already answered by N6. However, N5 updates N3 with the newly inserted event before N3 replies at N4 with the query result, therefore, the new event will be included in such result. In order to decrease this effect, we bound the number of hops a zone can be replicated away from its original owner to a limited number of hops.

We now present simulation results showing the ability of ZP/ZPR to decompose query hot-spots of different sizes.

5 Experimental Evaluation

In order to measure the ZP/ZPR performance, we created a simulator for a typical sensor network applying DIM, as presented in [10]. We also simulated GPSR to be used as the routing protocol. Then, we added the ZP/ZPR functionality to such network to compare the effect of applying ZP/ZPR on top of DIM versus using pure DIM.

5.1 Experimental Testbed

In our simulation, we tried to use the same experimental setting used in [10] to get similar DIM performance. Thus, we simulated networks of sizes ranging from 50 to 300 sensors, each having an initial energy of 100 units, a radio range of 40m, and a storage capacity of 10 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 selected a value of 2 for $threshold_1$. Concerning the PC parameters, we chosen a value of 0.3 for the E_1 and E_2 constants of equations (2) and (3), respectively. For the accessing frequency equations, we set Q_1 to 3, Q_2 to 0.8, and we gave Q_3 an 0.2 value. Note that these are just typical values for such thresholds. For an exact performance of ZP/ZPR, extensive binary search among the different combinations of such constants is required and is part of our future work.

Our simulation modeled a network of temperature sensors. The range of possible temperature values was $[30, 70]$. The simulation consisted of two alternating phases: *the insertion phase* and *the querying phase*. In the insertion phase, each sensor initiates (i.e., reads) 1 event and forward such reading to its owner sensor node. Event values were picked uniformly at random from the range of possible temperatures. The insertion phase was followed by a *querying phase* that is supposed to form a single query hot-spot within the network. In such phase, each sensor generates 4 queries of the form:

```
select NodeID, timestamp
from sensors
where temrature > p and temperature < q
```

In order to model the worst case scenario, queries initiated by all sensors used the same $[p, q]$ range, thus, all queries were asking for events stored by the set of sensors responsible of the $[p, q]$ range. We alternate both phases for 5 consecutive times. At each round, the $[p, q]$ range is chosen uniformly at random from the range of possible temperatures. This *moving query hot-spot* effect was needed in order to validate the robustness of our algorithms and their ability of quick detection and decomposition of query hot-spots.

We ran the above simulation for various hot subrange sizes (from 0.05% to 10% of the total possible temperature
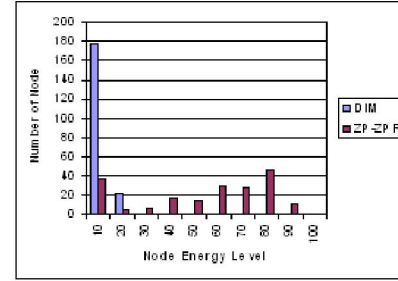

Figure 5. Node energy level distribution for a network with 200 nodes and 0.33% hot-spot
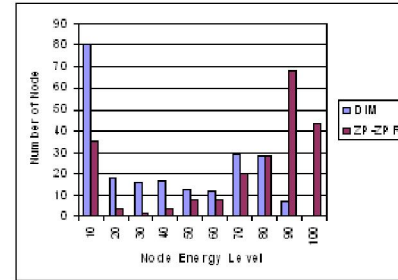

Figure 6. Node energy level distribution for a network with 220 nodes and 2.5% hot-spot

range). Note that the size of the hot subrange determines which of the two algorithms is applied more. ZP is used more for larger subranges while the usage of ZPR dominates as long as the hot subrange size decreases. Throughout this section, an (x, y) network is a network with x nodes and a hot subrange of size equal to $y\%$ of the whole attribute ranges. Note that y is also used to indicate the hot-spot size.

5.2 Results

The results of the simulations are shown in the Figures 5 to 9. In these Figures, we compare the performance of the pure DIM versus that of DIM appended with ZP/ZPR, based different performance measures. Note that we only report some of our findings due to space constraints.

Energy Consumption: Figures 5 and 6 present a clustering of sensor nodes into groups based on their energy level for $(200, 0.33)$ and $(220, 2.5)$ networks, respectively. In both Figures, the x-axis indicates is partitioned into 10 chuncks of size 10 each (recall that the initial node energy is 100 units), while the y-axis indicates the number of sensor nodes falling in each chuck, i.e. whose final energy level belongs to the energy range of such chunck. In both runs, almost all nodes fall in the first chuck for DIM, that is, all nodes have energy $\in [0, 10]$. Thus, a query hot-spot causes the death, or closeness to death, of the majority of the network nodes in the pure DIM. By running ZP/ZPR on
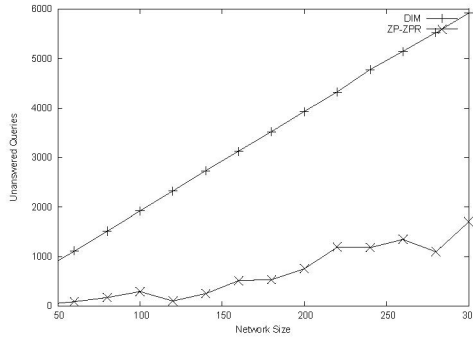
Figure 7. The number of unanswered queries for networks with a 5% hot-spot

Figure 8. Average node storage level for networks with a 0.05% hot-spot

Figure 9. Number of full nodes for networks with a 0.05% hot-spot

top of DIM, nodes tend to be more uniformly distributed among chuncks. Thus, we achieve a better balancing of the energy consumption among the sensor nodes, which leads to more energy savings and less node deaths. Note that this also shows that applying ZP/ZPR does not cause an additional energy overhead on the sensor nodes.

Quality of Data: Figures 7 shows the number of unanswered (failed) queries for networks with 5% hot-spots. It is clear that ZP/ZPR algorithms improve the QoD by decreasing the number of dead nodes (based on the previous result), which leads to loosing less stored data, thus, decreasing the number of unanswered queries. Note that the gap between DIM and ZP/ZPR, in terms of the number of unanswered queries, increases with the increase of the network size.

This result has a very important implication on the *data accuracy* of the sensor readings output from a network experiencing a hot-spot. The success of the ZP/ZPR combination in decomposing query 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 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 scheme.

Load Balancing: Figure 8 presents the average node storage level for networks with an 0.05% hot-spot. The Figure shows the average node load is almost the same for both DIM and ZP/ZPR. Recall that, in our insertion phases, each sensor generated 5 events. As events are uniformly distributed, the average storage of a sensor should not exceed 5 events in the pure DIM. Although ZP/ZPR moves events away from their original owners, the Figure shows that it achieves the same average storage per node as DIM, thus, does not disturb the uniformity of the events among sensors. Figure 9 presents the number of full nodes for net-
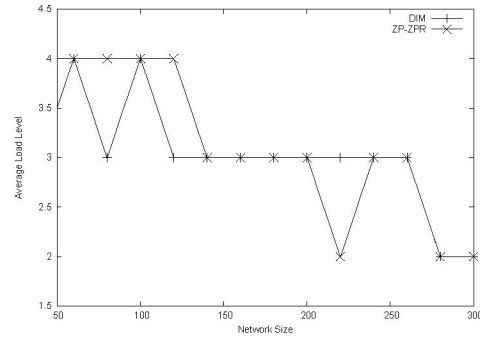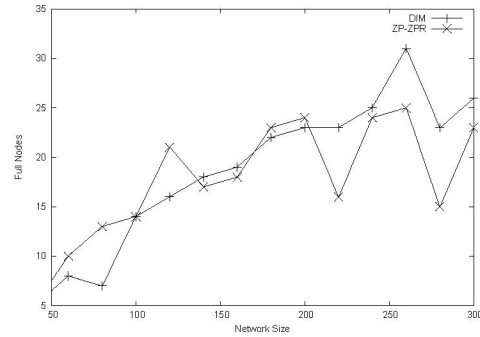
works with an 0.05% hot-spot. By *full nodes*, we mean the nodes having full memories. In our simulation, this means a node having 10 events in its cache. The Figure shows that the performance of networks applying ZP/ZPR on top of DIM is equal to those applying the pure DIM scheme for smaller network sizes. ZP/ZPR algorithms perform better than pure DIM for larger network sizes.

Additionally, we have not observed any dropping of events due to storage overflow while appending DIM with ZP/ZPR. This means that the ZP/ZPR combination does not cause the formation of storage hot-spots by moving a large number of events to few sensor nodes, thus, increasing event droppings [1]. In conclusion, the ZP/ZPR algorithms improve the sensor network ability to support a higher portion of the hot-spot events, while achieving a better balancing of such events among sensor nodes.

6 Related Work

Many approaches have been presented in literature defining how to store the data generated by a sensor network. In the early age of sensor networks research, the main storage trend followed consisted of sending all the data to be stored in base stations, which lie within, or outside, the network. However, this approach may be more appropriate to answer

continuous queries, which are queries running on servers and mostly processing events generated by all sensors over a large period of time [5, 11, 19, 15, 3, 12, 13, 7].

To improve the *QoD* of ad-hoc queries, In-Network Data-Centric Storage (INDCS) techniques have been proposed [16]. These INDCS schemes differ from each other based on the events-to-sensors mapping used. The mapping was done using hash tables (and mapping based on event types) in DHT [16] and GHT [14], or using k-d trees in DIM [10].

The formation of query hot-spots due to irregularity, in terms of *sensor deployment* or *query load distribution*, represent a vital issue in current INDCS techniques [6]. Some possible solutions for *irregular sensors deployments* were highlighted in [6], such as routing based on virtual coordinates, or using heuristics to locally adapt to irregular sensor densities. To our knowledge, no techniques have been provided to cope with irregular query load distribution nor with query hot-spots in INDCS. A complentary work to our paper is that on exploting similarities in processing queries issued by neighboring sensors in a DCS scheme [17].

Recently, some load balancing heuristics for the *irregular events distribution* problem, namely the *Storage Hot-Spots* problem, were presented by [1, 9, 2]. A storage hot-spot arises in case a large percentage of the readings has similar, or close, values. Thus, it is basically depending on the sensed environment, as well as the sensed phenomena unlike query hot-spots that mainly depend on the ad-hoc queries, sent by first responders for example. Zone Sharing has been presented by [1] to locally decompose storage hot-spots. On the other hand, global and periodic load balancing of the k-d tree was adopted in [9, 2] to solve the storage hot-spots problem. It is clear that storage hot-spots are independent from query hot-spots. Thus, an algorithm decomposing one of them is not by any means guarranteed to decompose the other type. It is not clear for us how to extend any of the previously presented storage hot-spots decomposition algorithms to cope with query hot-spots.

7 Conclusions

In this paper, we presented two novel algorithms for decomposing query hot-spots in Data-Centric Storage sensor networks, Zone Partitioning (ZP) and Zone Partial Replication (ZPR). ZP is based on partitioning the hot zone storage responsiblity among larger number of sensors, while ZPR is based on replicating the hot zone in sensor neiboring the hot-spot area. Our experimental results show that applying the ZP/ZPR algorithms on top of the DIM scheme achieves good performance in decomposing query hot-spots of different sizes. This improves the QoD and increases energy savings, which implies an ameliration of the profit gained from the sensor network throughout the network lifetime.

References

[1] M. Aly, N. Morsillo, P. K. Chrysanthis, and K. Pruhs. Zone Sharing: A hot-spots decomposition scheme for data-centric storage in sensor networks. In *DMSN*, 2005.

[2] M. Aly, K. Pruhs, and P. K. Chrysanthis. Load balancing of in-network data-centric storage in sensor networks. Technical Report TR-06-133, Department of Computer Science, University of Pittsburgh, 2006.

[3] J. Beaver, M. A. Sharaf, A. Labrinidis, and P. K. Chrysanthis. Power-aware in-network query processing for sensor data. In *HDMS*, 2003.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. In *CACM, 18(9)*, 1975.

[5] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *MDM*, 2001.

[6] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin. Coping with irregular spatio-temporal sampling in sensor networks. In *HotNets-II*, 2003.

[7] H. Gupta, V. Navda, S. R. Das, and V. Chowdhary. Efficient gathering of correlated data in sensor networks. In *MobiHoc*, 2005.

[8] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless sensor networks. In *ACM Mobicom*, 2000.

[9] X. Li, F. Bian, R. Govidan, and W. Hong. Rebalancing distributed data storage in sensor networks. Technical Report No. 05-852, Computer Science Department, University of Southern California, 2005.

[10] X. Li, Y. J. Kim, R. Govidan, and W. Hong. Multi-dimensional range queries in sensor networks. In *ACM SenSys*, 2003.

[11] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, 2002.

[12] S.-J. Park, R. Vedantham, R. Sivakumar, and I. F. Akyildiz. A scalable approach for reliable downstream data delivery in wireless sensor networks. In *MobiHoc*, 2004.

[13] T. Pham, E. J. Kim, and W. M. Moh. On data aggregation quality and energy efficiency of wireless sensor network protocols. In *BROADNETS*, 2004.

[14] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govidan, and S. Shenker. GHT: A grographic hash table for data-centric storage. In *WSNA*, 2002.

[15] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. TiNA: A scheme for temporal coherency-aware in-network aggregation. In *MobiDE*, 2003.

[16] S. Shenker, S. Ratnasamy, B. Karp, R. Govidan, and D. Estrin. Data-centric storage in sensornets. In *HotNets-I*, 2002.

[17] P. Xia, P. K. Chrysanthis, and A. Labrinidis. Similarity-aware query processing in sensor networks. In *WPDRTS*, 2006.

[18] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *SenSys*, 2003.

[19] Y. Yao and J. Gehrke. Query processing for sensor networks. In *CIDR*, 2003.