Distance Sensitive Snapshots in Wireless Sensor Networks

Vinodkrishnan Kulathumani and Anish Arora

Department of Computer Science and Engineering, The Ohio State University
(vinodkri, anish)@cse.ohio-state.edu

Abstract. Global state snapshots are a fundamental primitive for wireless networks that sense and control real environments. Consistent and timely snapshots are potentially costly. Cost reduction is often realized by gathering only a "delta" from previous snapshots. In this paper, we explore an alternative form of efficiency by generalizing the notion of a snapshot to satisfy *distance sensitivity* properties, wherein the state of nearby nodes is available with greater resolution, speed, and frequency than that of farther away nodes. Our algorithms are memory efficient and do not require global time synchronization or localization.
For pedagogical reasons, we describe our solutions for the case of perfect 2-d grid topologies first, and then show how to extend them for higher dimensions, for network with irregular density, arbitrary sized holes, networks and non unit disk radios. We also discuss how different control applications can exploit these generalized snapshots.

1 Introduction

Sensor networks have found significant adoption in continuous observation applications and are now progressively being incorporated in distributed control applications, for instance, pursuer evader tracking [1,2] and control of distributed parameter systems such as flexible structures [3, 4]. These applications often require information from network nodes to be periodically delivered to one or more observer/controller nodes in the network in a consistent and timely manner. For example, in pursuer evader tracking, pursuer objects require ongoing knowledge of other pursuer/evader locations in order to maintain an optimum assignment. In distributed vibration control of flexible structures, controllers need to (re)estimate the modes of vibration using samples from across the network in order to optimally assign controllers for each mode and to use the optimal control parameters. Thus global state snapshots are fundamental for wireless networks that sense and control real environments.
Although consistency, timeliness, and reliability have traditionally been the main design considerations for periodic snapshots, their efficiency becomes essential when considering resource constrained wireless sensor networks. The standard way to realize efficiency is to communicate the "delta" from previous readings or from model-driven predictions, possibly in compressed form. In this paper, we explore a complementary form of efficiency based on the observation that many applications can accommodate generalized forms of snapshots, wherein the information delivered across the network is not necessarily consistent but satisfies certain *distance sensitive* properties: The state of nearby nodes has greater resolution (*distance sensitive resolution*), arrives faster (*distance sensitive rate*) and

with higher speed (*distance sensitive latency*). By way of example, consider: (1) In pursuer evader tracking, information about nearer objects are required at a faster rate and lower latency that that of farther objects for guaranteeing optimal pursuit [1,5]. (2) In scale based control [6] used for vibration control of flexible structures, different controllers are assigned to different modes (frequencies) of vibration; in this case, estimating characteristics of lower frequencies requires information from a wider area but that can be sampled at a slower rate and coarser resolution than that for nearer areas.

While collecting snapshots at a central base station has been a common pattern in sensor networks, delivering snapshots to nodes in-network is desirable from an efficiency and correctness perspective in large scale networks used for applications such as object tracking [5] and distributed control and is also an emerging pattern in applications involving mobile users. In this paper we focus on in-network delivery of snapshots.

Informal problem statement: Given is a connected wireless sensor network with N nodes embedded in an f dimensional space. Each node periodically generates m bits of information, can communicate at W bits per second, and is memory constrained.

> Design efficient snapshots of the network state that are distance sensitive in resolution, latency, and rate for periodic delivery at (some or all) nodes.

Contributions: In this paper, we systematically design wireless sensor network algorithms that periodically deliver distance sensitive snapshots to all nodes in the network. Our algorithms are easily adapted to allow snapshots to be delivered only to a subset of nodes as opposed to all nodes. They are memory efficient, requiring only $O(3^f * log(N^{\frac{1}{f}}) * m)$ bits per node. They are readily realized in networks with irregular density, networks with arbitrary sized holes, imperfect clustering, and non unit disk radios. We quantify the maximum rate at which information can be generated at each node so that snapshots are periodically delivered across the network, the algorithms can of course be operated at lower rates than these. For our services, global time synchronization is not required; a local notion of time however is needed to ensure fair scheduling of transmission of nodes.

Overview of algorithms and main results: Consider an ideal network where nodes are embedded in a virtual 2-d grid such that there is exactly one node at each grid location and that each grid node can reliably reach each of its neighbors in the grid and no others. Snapshots with distance sensitive latency may be realized in these grids, firstly, by scheduling each node to transmit its local view of the network so as to not interfere with its neighbors and, secondly, by ensuring that the schedules all nodes to transmit at the same rate. In order to ensure uniform latency, we introduce a single level of clustering to regulate the flow of information in all directions by proceeding in rounds. Intuitively, a round is a unit of time when information is exchanged between any level 1 cluster and all its neighboring clusters. Our scheduling and other protocol actions at each step are such that information is propagated across the network in a pipelined

manner; by this, new information can be generated at a node as soon as previous information has been dispersed only to its local neighborhood as opposed to the entire network.

> In this first algorithm, in a snapshot S of the network delivered to all nodes the staleness of the state of a node i in S is $O(3^f * N * m * d)$, where $d = dist(i, j)$, and the average network communication cost is as high as $O(N^2 * m)$ for N samples (one from each node).

To add distance sensitive resolution, instead of dispersing the individual state of each node, we map the state of nodes into aggregate values of non-overlapping regions. We then deliver snapshots across the network such in a snapshot delivered to a node j, the size of a region into which a node i is mapped increases as $dist(i, j)$ increases. Thus, the resolution with which i is represented in the snapshot decreases as $dis(i, j)$ increases. To achieve this kind of snapshot delivery, we refine the clustering of nodes into a hierarchical one with a logarithmic number of levels as the network size. The basic idea is that a clusterhead at each level compresses data from all nodes in that level into m bits. Thus, the data aggregated at each level is represented by the same number of bits. At higher levels, the data is summarized with a coarser resolution as these levels contain more nodes.

> In this second algorithm, in a snapshot S of the network delivered to node j the resolution of the state of a node i in S decreases as $O(d^f)$, the staleness of the state of a node i in S is $O(3^{2f} * m * log(n) * d)$ and the average network communication cost for N samples is $O(3^f * log(n) * N * m)$.

To achieve distance sensitive rate, we schedule the delivery of aggregated information at each level such that information of higher levels is delivered over a larger interval as opposed to lower levels. We do this in two ways. In the first solution, we allocate an exponentially increasing number of bits per message to lower level aggregates so that they are delivered at a faster rate. In the second solution, we allocate more time for aggregation and dispersion of lower level data.

> In the first of these two algorithms, the average communication cost per N samples (one from each node) is $O(3^f * N * (m + log(n/m)))$. In the second, the average communication cost per N samples (one from each node) in the second algorithm is $O(N * m)$, but the staleness of the received states grows as $O(d^f)$.

Our algorithms allow for a user-pluggable aggregation function. We require only that the function, say f, be idempotent and satisfy the following *decomposability* property: $\forall a, b$, $f(a \cup b) = f(f(a) \cup f(b))$. Examples of such functions are average, max, min, count and wavelet functions.

We then relax our regularity assumptions and describe how our algorithms handle the cases of non uniform density, non uniform radio range and holes of arbitrary sizes in the network. The case of over density is modeled as certain virtual grid locations containing more than 1 node. In the case of holes in the

network, we show that our algorithms achieve distance sensitivity in terms of the shortest communication path between any two nodes as opposed to the physical distance.

Outline of the paper: In Section 2, we present the system model. In Section 3, we design a snapshot service that has the property of *distance sensitive latency*. In Section 4, we design a snapshot service that has the additional property of *distance sensitive resolution*. In Section 5, we refine our snapshot service so that snapshots are delivered with a *distance sensitive rate* property. In Section 6, we consider irregular networks. We discuss related work in Section 7 and make concluding remarks in Section 8.

2 Model and specification

In this section, we present the system model and a generalization of the concept of snapshots based on distance sensitive properties.

Network model: A sensor network consists of N nodes that are embedded in an f-dimensional space. We let n abbreviate $N^{\frac{1}{f}}$. The nodes induce a connected network where each communicate at W bits per second. Nodes are synchronized in time. Each node j periodically generates m bits of (sensor) information, and maintains a data structure comprising the most recent state of nodes (or partitions of nodes) and a timestamp associated with that state.

In the next three sections (3-5), for ease of exposition, we restrict our attention to sensor networks that form a 2 dimensional grid with a node at every grid location. We further assume that node communication follows an idealized disk model: specifically, each node can communicate reliably with all its neighbors in the grid and with no others. We define the neighbors of node j to be the ones to its north, east, west, and south and also to its northeast, northwest, southeast, or southwest that exist in the grid; we denote these (up to 8) neighbors as $j.n$, $j.e$, $j.w$, j, s, $j.ne$, $j.nw$, $j.se$ and $j.sw$ respectively. In Section 6, we remove each of these restrictive assumptions.

Definition 1 (Snapshot S). *A snapshot S is a mapping from each node in the network to a state value and a timestamp associated with that state value.*

A consistent snapshot is one where the timestamps associated with each state value are all the same. The *staleness* of a state value in S is the time elapsed between its timestamp and the current time. We now consider a generalization where state values do not necessarily correspond to the same instant of time but their staleness enjoys a distance sensitive property.

Definition 2 (Snapshots with distance sensitive latency). *A snapshot S received by a node j has* distance sensitive latency *if the staleness in the state of each node i in S decreases as $dist(i, j)$ decreases.*

We now further generalize the notion of snapshots so that state is associated with partitions p of the network as opposed to individual nodes. Let P be a partitioning of the network.

Definition 3 (Snapshot S of P). *A snapshot S of P is a mapping from each partition p in P to a state value and a timestamp associated with that state value.*

The generalized definition is useful even if P is not a total but a partial partition, i.e., not all nodes are represented in the snapshot. The state and timestamp of each p in S intuitively represent the aggregate state of all nodes in the partition and the aggregate timestamp. We assume that the timestamp of recording the state of all nodes in any partition p is the same, and refer to this common value as the aggregate timestamp. Note that the aggregate timestamp of different partitions may be different.
As there may not exist a mapping from the aggregate state of a partition to the exact state of individual nodes that was recorded for the purpose of computing the aggregate, the latter may be estimated using some function of the state of the partition. The *resolution* of the state of a node in a snapshot is an inverse measure of the error between the state of the node that was recorded and the aggregate state of the partition p that it belongs to.
We are interested in snapshots where the increase in the error in the state of a node is bounded by the size of the partition p to which it belongs. This leads us to consider a generalization where the resolution of the state of a node increases as distance decreases.

Definition 4 (Snapshots with distance sensitive resolution). *A snapshot S of P received by a node j has* distance sensitive resolution *if the resolution of the state of each node i covered by P increases as $dist(i, j)$ decreases.*

Informally speaking, the size of the partition to which the state of node i is mapped into in a snapshot received at j increases as $dist(i, j)$ increases. Therefore the resolution with which i is represented in S decreases with distance.
Finally, we consider a generalization where the rate at which state of the nodes is reported to a node decreases as the distance of the nodes increase.

Definition 5 (Snapshots with distance sensitive rate). *A node j receives snapshots of P with* distance sensitive rate *if the rate at which the state of each node i covered by P is updated in snapshots received by j increases as $dist(i, j)$ decreases.*

3 Distance sensitive latency snapshots

In this section, we describe a snapshot service that has distance sensitive latency. Moreover, by introducing a single level of clustering, we also achieve information flow with uniform latency in all directions. Uniformity is a desirable property especially when aggregation needs to be performed.
Clustering: In order to achieve uniform latency, we create a single level of clustering. The grid is partitioned in 3 by 3 sub-grids of nodes, with the center node in each sub-grid cluster being its clusterhead. We call the clusterhead a level 1 node and the rest of the nodes in the cluster as level 0 nodes. This kind of clustering is illustrated in Fig. 1.
Schedule: We schedule the nodes to transmit in rounds. A round is a unit of time in which information is exchanged between a level 1 clusterhead and all of its 8 neighboring level 1 clusterheads. Each round is divided into multiple slots. In the first slot, all level 1 clusterheads transmit. In the remaining slots,

all level 0 nodes in each cluster transmit twice. The second transmission by a node within a round takes place after all its 8 neighbors have transmitted at least once. Intuitively speaking, during the first turn for a node, information is communicated outwards from the clusterhead. In the next turn for the node, information is communicated to the level 1 clusterhead that the node belongs to. A simple non-interference schedule that satisfies these constraints is one where all level 0 nodes take turns in a round robin manner. Each round thus consists of 17 slots.

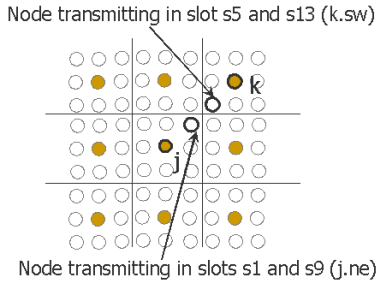

Fig. 1. 1 level clustering

Algorithm $S1$**:** In slot 0 of every round, the level 1 nodes update their own state in the local data structure and transmit the entire data structure. The level 0 nodes in each cluster update their local data structure as follows: wlog, node $j.ne$ copies the state of all nodes in its own cluster and the state of nodes in all level 1 clusters that are not north east of j.

To explain the actions in other slots, without loss of generality, consider level 1 nodes j and k and level 0 nodes $j.ne$ and $k.sw$, as shown in Fig. 1.

- In the first slot for node $j.ne$, $j.ne$ transmits its local data structure which contains the updates that were heard from j. Node $k.sw$ updates the state of all nodes in clusters that are southwest of k.
- In the second slot for node $j.ne$, $j.ne$ transmits its local data structure which contains the updates sent from k and $k.sw$, heard via $k.sw$. Node j updates the state of all nodes in clusters that are north east of j. □

In the remaining slots, the states are exchanged along the other axes around j. In algorithm $S1$, information flows between any 2 nodes through paths defined by level 1 clusters. Moreover, by the rules of updating a unique path is maintained for communicating state from a node to any other node [7]. Within a round, information is fully exchanged in a level 1 neighborhood. Thus, the latency involved in moving information between a pair of nodes depends on the number of level 1 clusters in their path, and this is uniform in all directions. Note also that between a pair of level 1 nodes, information is exchanged in 17 slots and the length of the path through level 1 nodes is proportional to d. We now state the following lemmas, the proofs of which have been relegated to the technical report for reasons of space.

Lemma 1. *In* $S1$*, the maximum staleness in the state of a node* i *received by a snapshot at node* j *is* $O(N * m * d)$ *where* $d = dist(i, j)$. □

Lemma 2. *In* $S1$*, the average communication cost to deliver a global snapshot to all nodes per sample from each node is* $O(N^2 * m)$. □

4 Distance sensitive resolution snapshots

To incorporate the property of distance sensitive resolution, we refine the partitioning of the network into a hierarchical one with a logarithmic number of

levels, which are numbered $0..(log_3 n)$. A 3 by 3 set of 9 level r clusters form a cluster at level $r+1$, as illustrated in Fig. 2. Each node belongs to one cluster at each level, and each cluster has a clusterhead which is the center node of that cluster. A clusterhead at level r is also a clusterhead at levels $0..r-1$.

Overview of algorithm $S2$: The basic idea is that a clusterhead at each level compresses data from all nodes in that level into m bits. Thus, aggregated data at each level is represented by the same number of bits. At higher levels, data is summarized into a coarser resolution as the levels contain more nodes. The aggregated data is then dispersed to all nodes at that level. This solution suffers from a multi-level boundary problem however: two nodes could be neighbors but belong to a common cluster only at level $r \gg 1$. Thus despite being neighbors, both nodes get a summary of the other at a much coarser resolution than desired. The multi-level boundary problem is illustrated in Fig. 2, where nodes j and k are neighbors at level 0 but belong to a common cluster only at level 3. To avoid this problem, we disperse a summary computed at level r not only to nodes in level r cluster, but also to nodes in all neighboring level r clusters. We now describe a pipelined implementation of this algorithm.

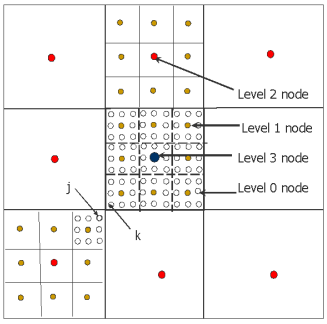

Fig. 2. Hierarchical clustering

Notations: Let $j.L$ be the highest level for which j is clusterhead. Note that there are at most 8 neighbors at each level for each node in the grid topology. We implement virtual trees along the structure at each level. To describe these trees, we will need the following definitions.

Definition 6 ($tree(k,j)$)**.** *$tree(k,j)$, where j is a level k clusterhead, is a level k tree formed with j as root and spanning all nodes in the level k cluster of j and all level k clusters that are its neighbors.*

Definition 7 ($j.in(k,y)$)**.** *For each $tree(k,y)$ that j belongs to, $j.in(k,y)$ is j's parent towards root y.*

Definition 8 ($j.out(k,y)$)**.** *For each $tree(k,y)$ that j belongs to, $j.out(k,y)$ is the set of j's descendants on the tree.*

Definition 9 ($M(k,y)$)**.** *$M(k,y)$ is the level k summary computed by a level k clusterhead y.*

In Fig. 3, a level 1 tree rooted at j is shown as an illustration. The level 1 tree extends up to all level 0 nodes in its own cluster and level 0 nodes in the 8 neighboring level 1 clusters. The trees represent the distance up to which an aggregate at any level is propagated.

Schedule: In the first slot of a round, level 1 clusterheads transmit. In the remaining slots, all the level 0 nodes per cluster take turns and transmit twice such that the second transmission occurs after all its 8 neighbors have transmitted at least once, as described in algorithm $S1$.

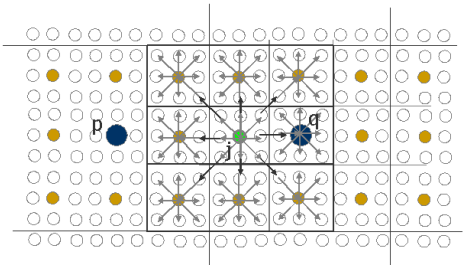

Fig. 3. Illustrating level 1 tree rooted at j

Local storage: Each node i stores the most recent value of $M(x, y)$ received by i for each $tree(x, y)$ that i belongs to. The state of any node j is obtained as a function of $M(x', y')$ where x' is the smallest level that contains information about j. Recall that the resolution of the state of j decreases as the number of nodes in the aggregate $M(x', y')$ increases.

Algorithm $S2$**:** We describe the actions executed by the nodes.

- In slot 0 of each round nodes with $j.L > 0$ compute the summary $M(r, j)$ for each level $1 \leq r \leq j.L$ that they are a clusterhead of based on the corresponding lower level information received in the previous round. The computed summary at each level is transmitted to the children on the respective tree rooted at j. Thus $M(r, j)$ is sent to $j.out(r, j)$ for $1 \leq r \leq j.L$.
- To explain the actions of level 0 nodes, without loss of generality, consider level 1 nodes j and k and level 0 nodes $j.ne$ and $k.sw$ as shown in Fig. 1.
  - In first slot for $j.ne$, for each $tree(x, y)$ that $j.ne$ belongs to but is not a leaf of, transmit $M(x, y)$ as heard in slot 0 from $j.in(x, y)$ to $j.out(x, y)$. Also, transmit its own information $M(0, j.ne)$ to children in the level 0 tree rooted at $j.ne$.
  - In second slot for $j.ne$, for each $tree(x, y)$ that $j.ne$ belongs to but not a leaf of, transmit $M(x, y)$ as heard in slots 2 to 8 from $j.in(x, y)$ to $j.out(x, y)$.
- The action at any node j upon receiving a message from i is as follows: for each $tree(x, y)$ that j belongs to, store $M(x, y)$ if $i = j.in(x, y)$. □

In summary, aggregates computed at each level are copied only going downwards along a tree. This is sufficient for a level r node to compute aggregates from level $r-1$ nodes, because a tree at level $r-1$ extends up to all level 0 nodes in neighboring level $r-1$ clusters. And one of the neighboring level $r-1$ node is a level r node. Thus, when a computed aggregate by any node is being dispersed to nodes in its own cluster and the neighboring clusters, it is also being sent *in* to a higher level node to compute an aggregate. In Fig. 3, nodes p and q are level 2 clusterheads. Note that the level 1 tree rooted at j reaches the level 2 clusterhead q that j belongs to. Since a level r node is equidistant from all level $r-1$ nodes and because of the uniform latency property, the computed summaries are synchronous.

Lemma 3. *In* $S2$*, the slot width* s_W *needed is* $\frac{(9*log(n)-7)*m}{W}$ *bits per second.*

Proof. Note that, at most 9 trees at levels $1..log(n)-1$ can pass through each node. There is only one level $logn$ tree. Also j belongs to only one level 0 tree for which it is not a leaf. Hence the maximum message length needed per slot is $(9 * log(n) - 7) * m$ bits [7]. The result follows. □

Lemma 4. *In $S2$, the maximum staleness in the state of a node i received by a snapshot at node j is $O(log(n) * m * d)$ where $d = dist(i, j)$.*

Proof. Consider a node p at level r. To compute a summary at level r, level $r-1$ summaries are needed. $dist(p, q) = 3^{r-1}$, where q is any node in the set $p.nbr(r)$. A level $r-1$ summary is computed based a level $r-2$ summary, and so on until level 0. Upon summation, the staleness of a level 0 (individual node) state information in a level r summary is equal to $(17/2) * 3^{r-1} * s_w$ [7]. The maximum distance traveled by a level r summary is $(3/2) * 3^r$ with latency bounded by $(17/2) * 3^r * s_w$. The minimum distance between j and i for which a level r summary is the smallest level that contains information about j is 3^{r-1}. The result follows. □

Lemma 5. *In $S2$, the resolution of state of a node i in a snapshot received at node j is $\Omega(\frac{1}{d^2})$ where $d = dist(i, j)$.*

Proof. In a level r summary, the state of 9^r nodes is compressed into m bits. We thus regard the error in the state of each node in that summary to be $O(9^r)$. The minimum distance between i and j at which j gets a level r summary of i but not a level $r-1$ summary of i is 3^{r-1}. Thus, the error in the state of i in a snapshot received at j is $O(d^2)$ and the resolution of state of i in a snapshot received at j is $\Omega(\frac{1}{d^2})$, where $d = dist(i, j)$.

Lemma 6. *In $S2$, the average communication cost in the network to deliver a snapshot of one sample from each node to all nodes is $O(N * log(n) * m)$.*

Proof. To deliver a snapshot with a sample from each node, every node communicates $O(m * log(n))$ bits n times. And to deliver a snapshot with y samples from each node, every node communicates $O((n + y) * (m * log(n)))$ bits, since all the y samples are pipelined. Hence, if y is large and $y = \Omega(n)$, the average communication cost at each node to deliver a snapshot of a sample from each node to all nodes is $O(m * log(n))$. The average communication cost over N nodes is $O(N * (m * log(n))$. □

Lemma 7. *In $S2$, the memory requirement per node is $O(log(n) * m)$ bits.*

Proof. Recall that the data structure maintained at each node is the most recent value of $M(x, y)$ received by i for each $tree(x, y)$ that i belongs to. Nodes do not buffer information to be forwarded over multiple rounds. The maximum number of trees through any node is $O(log(n))$, with m bits of information flowing along each tree. The result follows.

Extending to other dimensions: In an f dimensional structure, nodes are divided into clusters with 3^f nodes per cluster. Thus there are $3^f - 1$ level 0 nodes per cluster. Each round consists of $2 * 3^f - 1$ slots and thus the number of slots per round increases proportional to 3^f. Further, there can be at most $3^f - 1$ neighbors at each level. Thus, there can be $O(3^f * log(n))$ trees passing through each node. Using these, we can generalize Lemmas for performance of $S2$ [7]. We summarize our results for all algorithms in Fig. 4 in Section 5.

5 Distance sensitive rate snapshots

In this section, we describe two algorithms in which nodes receive snapshots that are distance sensitive in latency, resolution and also distance sensitive in rate.

5.1 Distance sensitive rate by data division

We partition the network hierarchically into clusters and schedule nodes to transmit in rounds exactly as we did in algorithm $S2$. However, instead of transmitting m bits for each level of data in every round, we allocate the number of bits hierarchically. Accordingly, a message transmitted by a node in any given round consists of m bits for each level 0 information, $m/3$ bits for each level 1 information, and 1 bit for each level from $log(m)$ to $log(n)$.

Algorithm $S3a$**:** By way of refining algorithm $S2$, consider a level 0 node with $j.L = r$. A level r summary is computed by this node once every 3^r rounds based on the most recent level $r-1$ summaries it receives. This summary $M(r, j)$, which consists of m bits, is transmitted in slot 0 of each round with $max(1, \frac{m}{3^r})$ bits per round. Thus, a level r summary is sent over $min(3^r, m)$ rounds. The actions for forwarding nodes remain the same except for the change that each node now only receives a fraction of $M(x, y)$ in every round for each $tree(x, y)$ that it belongs to, and it forwards only that fraction in the next round. We now state the latency and communication cost of algorithm $S3a$.

Lemma 8. *In $S3a$, the maximum message length needed per slot in algorithm is $11 * \frac{m}{2} + 9 * log(\frac{n}{m})$ bits.* □

Lemma 9. *In $S3a$, the maximum interval between when a node j receives the state of node i is $O((m + log(n/m) * d)$, where $d = dist(i, j)$.* □

Lemma 10. *In $S3a$, the maximum staleness in the state of a node i received by a snapshot at node j is $O((m + log(n/m)) * d)$ where $d = dist(i, j)$.* □

Lemma 11. *In $S3a$, the average communication cost to deliver a snapshot of one sample from each node to all nodes is $O(N * (m + log(n/m))$.* □

5.2 Distance sensitive rate by time division

Again, we hierarchically partition the network into clusters and schedule nodes to transmit in rounds exactly as in algorithm $S2$.However, instead of allocating exponentially increasing number of bits per level in each round, we allocate each round to a particular level and the information corresponding to that level is propagated only in that round. The frequency at which a round is allocated to a particular level increases exponentially as level decreases.

Algorithm $S3b$**:** Consider level $r > 0$. Let the rounds be numbered starting from 1. All rounds enumerated by $2^{r-1} + i \times 2^r$ for $i > 0$ are allocated to round r, where $r > 0$. A level 0 information is carried in all rounds. Consider a round s that belongs to level $r_s > 0$. A node j with level $j.L \geq r_s$ computes the summary only corresponding to level r_s. The computed summary at each level is transmitted to the children on the respective tree rooted at j. Level 0 nodes forward information only pertaining to level r_s in round s.

Lemma 12. *In S3b, the maximum message length needed per slot is* $10 * m$ *bits.*

Lemma 13. *In S3b, the maximum staleness in the state of a node i received by a snapshot at node j is* $O(m * d^2)$ *where* $d = dist(i, j)$. □

Lemma 14. *In S3b, the maximum interval between when a node j receives the state of node i is* $O(m * d)$. □

Lemma 15. *In S3b, the average communication cost to deliver a snapshot of one sample from each node to all nodes is* $O(N * m)$. □

Lemma 16. *In S3a and S3b, the memory requirement per node is* $O(log(n) * m)$.

Both algorithms $S3a$ and $S3b$ can be generalized to f dimensions just as algorithm $S2$ is [7]. We summarize all our results in Fig. 4.

Algorithm	Staleness	Communication cost	Resolution	Interval	Memory
$S1$	$O(3^f * N * m * d)$	$O(N^2 * m)$	$full$	independent of d	$N * m$
$S2$	$O(3^{2f} * log(n) * m * d$	$O(3^f * N * m * log(n))$	$\Omega(\frac{1}{d^f})$	independent of d	$3^f * log(n) * m$
$S3a$	$O(3^{2f} * (m + log(n/m)) * d)$	$O(3^f * N * (m + log(n/m)))$	$\Omega(\frac{1}{d^f})$	$O(3^f * (m + log(n/m)) * d)$	$3^f * log(n) * m$
$S3b$	$O(3^{2f} * m * d^2)$	$O(3^f * N * m)$	$\Omega(\frac{1}{d^f})$	$O(m * d * 3^f)$	$3^f * log(n) * m$

Fig. 4. Summary of results for snapshot algorithms

6 Irregular networks

In this section, we show how our algorithms continue to yield distance sensitive snapshots in the following cases: non uniform density, holes of arbitrary sizes within the connected network, non unit disk radios and imperfect clustering.

Clustering Model CM**:** We assume the existence of a clustering layer that partitions the general but connected network, as modeled in Section 2, into hierarchical clusters such that every network node belongs to one cluster at each level. As perfect (i.e., regular and symmetric) clustering may no longer be possible, we weaken that assumption to: each level 1 cluster includes all nodes that are 1 hop away but may also include nodes that are up to some bounded number of hops, z, from it. Likewise, all higher level clusterheads also have the same radius range as opposed to a uniform radius.

More formally, our clustering assumption is stated as follows. For simplicity we specify the model for a 2 dimensional network that can be generalized to f dimensions. In this model, we refer to distance in terms of communication *hop* distances.

- ($C1$) All nodes within hop distance $\frac{3^k-1}{2}$ from a level k clusterhead belong to that cluster.
- ($C2$) The maximum hop distance of a node from its level k clusterhead is $z^k \times \frac{3^k-1}{2}$.
- ($C3$) There exists a path from each clusterhead to all nodes in that cluster containing only nodes belonging to that cluster.
- ($C4$) At all levels $k > 0$, there is at least one and at most 8 neighboring level k clusters for each level k clusterhead and there exists a path between any two neighboring clusterheads.

We note that the existence of such clustering solutions has been validated in previous research [8] and also been used in the context of object tracking.

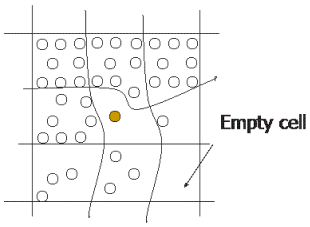

Fig. 5. Virtual grid

Once the network has been partitioned into clusters, we impose a virtual grid on the network, as shown in Fig. 5. Each level 0 node belongs to some cell, but now each cell in the virtual grid may contain any number of nodes. In particular, cells may be empty and empty cells may be contiguous; we call sets of contiguous empty cells the *holes* of the network.

Over density cells: In the virtual grid, each cell gets a slot to transmit as described in algorithm $S2$. When a cell has more than one node, each node in the cell gets a turn over multiple rounds to send its data, resulting in time sharing between nodes of a cell to transmit its own data. However, once data is sent out from the source, the forwarding of the data does not incur this extra delay despite going through denser cells. This is because any node in the dense cell that gets a turn in a given round can forward the data heard in the previous round from neighboring cells.

Under density cells, holes, and imperfect clustering: We first describe the changes needed in the scheduling to handle clusters of non uniform size. We then describe how distance sensitivity is preserved.

Scheduling scheme (FS): Recall that a *round* is a unit of time in which information is exchanged between a level 1 clusterhead and all its neighboring level 1 clusterheads. In the general model, a level 1 cluster can cover up to a z hop neighborhood. Accordingly, the basic round scheduling introduced in Section 3 is adapted to have $O(3^z)$ level 0 slots that fulfill the function of a round. Some slots may not be utilized because the cells may be empty.

Distance sensitivity: Recalling the clustering specifications stated above, consider any two nodes i and j in the network. Let the shortest path between these two nodes in the presence of holes be hop distance p.

Lemma 17. *Under clustering model CM, if k is the smallest level at which i and j are neighbors then $p > 3^{k-1}$.*

Proof. Note that i and j are not neighbors at level $k-1$. And if $p \leq 3^{k-1}$, then a level $k-1$ cluster cannot exist between i and j since from property $C1$, a level $k-1$ cluster has a minimum radius of $\frac{3^{k-1}-1}{2}$. □

Theorem 1. *Under model CM, algorithms $S2$, $S3a$ and $S3b$ yield snapshots that retain their distance sensitive properties.*

Proof. From the previous lemma, the minimum distance between two nodes i and j for which level k is the smallest level at which i and j are neighbors is 3^{k-1}.

Despite the fact the trees are not formed along the regular grid pattern, it still holds that not more than 9 trees per level pass through any node. This is because there at most 8 neighboring level k clusters for any level k cluster. Moreover, the maximum degree of any node in all trees is still 8, by imposing the virtual grid for level 0. Therefore, the slot width allocations in algorithms $S2$, $S3a$ and $S3b$ are sufficient to transmit all information. □

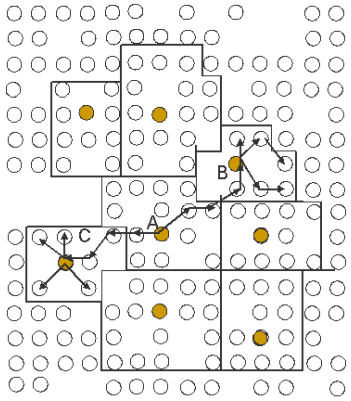

Fig. 6. Handling holes in dense networks

Fig. 6 illustrates how snapshots are communicated in irregular networks. The figure shows a level 1 cluster with a clusterhead A that has 7 neighboring level 1 clusters. The small unfilled circles represent cells of the virtual grid; these may contain one or more level 0 nodes. The level 1 clusters cover up to a 2 hop neighborhood. The figure also shows a level 1 tree rooted at A and extending up to clusters B and C.

Non-uniform radio range If communication range were relaxed to radio interference range varying from 1 to s hops, the basic scheduling for each round would need to take into account this additional interference. This would result in longer round lengths proportional to the size of interference region.

Implementation considerations: We now highlight considerations for implementing our snapshot services in wireless sensor networks. The snapshot services that we consider in this paper are high density operations and TDMA [9] is naturally suited for such scenarios as interference can be avoided. But we do not need global time synchronization in the network. Nodes in their network can learn their TDMA slots by knowing their relative position to that of a clusterhead and locally scheduling in a non interference manner. Note that our snapshot services are continuous and we do not recover a *lost* message. On the other hand we avoid message losses by interference free scheduling.

Another issue to consider is that of localization. For our snapshot services, information is communicated only along a tree structure that is rooted at clusterheads of different levels. Knowledge of location is not needed in the protocol actions; only knowledge of which trees a node belongs to is sufficient. Also it is sufficient for the nodes to be scheduled in a non interference manner, not particularly in any order. Thus localization is not required for our snapshot services.

7 Related work

Communicating periodic global state snapshots is a well studied problem in distributed systems [10] and consistency, timeliness and reliability have been the main design considerations in those studies. But efficiency becomes essential when considering periodic snapshots for resource constrained wireless sensor networks. To the best of our knowledge algorithms for delivering periodic snapshots across a wireless sensor network have not been studied before.

A common approach to achieving compression for efficiency is to exploit the temporal and spatial correlation of data being shared. For example, in [11], the authors propose a framework for a one time all-to-all broadcast of sensor data assuming the data is spatially correlated. Instead, in this paper we do not require data to be correlated. At the same time, our algorithms can be used in conjunction with other forms of compression.

Fractionally cascaded information [12] is a form of distance sensitive resolution that is widely used in computational geometry community for speeding up data structures. Recently, fractional cascading has been used for sensor networks as an efficient storage mechanism [13, 14]. Data is first stored at multiple resolutions across the network, which is then used to efficiently answer aggregate queries about a range of locations without exploring the entire area. In contrast, we have considered a model where information is generated and consumed on an ongoing basis. At the same time these services can be used in range based querying as well as in several other control applications.

An algorithm for creating the multi-resolution data structure based on probabilistic gossip mechanism has been discussed in [14]. In [14], the algorithm described is for a one shot dispersion and proceeds in stages while our services are for a model where information is consumed on an ongoing basis and accordingly we describe a pipelined implementation that is based on scheduling. In [14], the aggregation oprations are duplicate insensitive and global time synchronization is assumed while we do not require either of these properties. Our comunication costs and latency are lower than those in [14] and we also describe services that additionally have distance sensitive rate properties. But we note that while we assume hierarchical clustering in our solutions, the algorithm in [14] does not.

The idea of *distance sensitive rate* has also arisen in other contexts. Fisheye state routing is a proactive routing protocol [15] that reduces the frequency of topology updates to distant parts of the network.

Recently algorithms for bulk data collection in sensor networks have been proposed. In [16] data is collected from one node at a time, while [17] performs concurrent, pipelined exfiltration of data using TDMA schedules. Our algorithms can be specialized for the case of bulk convergecast and we additionally emphasize on efficiency using distance sensitive properties.

8 Conclusion

We have generalized the basic notion of snapshots using distance sensitive notions and accordingly designed efficient wireless sensor network algorithms that periodically deliver them. We achieve compression by forming hierarchical clusters and aggregating information at clusterheads. To communicate the snapshots, we embed logical trees rooted at clusterheads that extend up to all neighboring clusters at the corresponding level. Aggregate information at each level is then propagated downwards along the respective tree and this is sufficient for higher level clusterheads to compute respective aggregates. Our algorithm actions are such that information propagates in a pipelined manner; by this, new information can be generated as soon as previous information has been dispersed to a local neighborhood as opposed to the entire network. We achieve further compression in our algorithms by exponentially decreasing the bandwidth allocated to aggregates at higher levels.

Our algorithms are memory efficient and realizable in networks with irregular density, with arbitrary sized holes, and imperfect clustering. We have quantified the maximum rate at which information can be generated at each node so that snapshots are periodically delivered across the network; the algorithms can be

operated at lower rates. We have specified the allowable aggregation functions in abstract terms, allowable functions include average, max, min and wavelet functions. Our algorithms neither require global time synchronization nor localization.

We expect to implement our snapshot algorithms in the context of applications such as pursuer evader tracking and vibration control, and study their performance and tradeoffs more exhaustively in the future.

References

1. H. Cao, E. Ertin, V. Kulathumani, M. Sridharan, and A. Arora. Differential games in large scale sensor actuator networks. In *IPSN*, pages 77–84. ACM, 2006.
2. B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. Sastry. Distributed control applications within sensor networks. In *Proceedings of the IEEE*, volume 91, pages 1235–46, Aug 2003.
3. Challenge problem description for network embedded software technology (nest). Boeing Tech Report, The Boeing Company, St. Louis, MO 63166, April 2002.
4. Y. M. Kim, A. Arora, and V. Kulathumani. On Effect of Faults in Vibration Control of Fairing Structures. In *Fifth International Conference on Multibody Systems, Nonlinear Dynamics and Controls (MSNDC)*, 2005.
5. V. Kulathumani, M. Demirbas, and A. Arora. Trail: A Distance Sensitive WSN Service for Distributed Object Tracking. In *European Conference on Wireless Sensor Networks*, 2007.
6. K. Chou, D. Flamm, and G. Guthart. Multiscale Approach to the Control of Smart Structures. In *SPIE*, volume 2721, pages 94–105, 1996.
7. V. Kulathumani and A. Arora. Distance Sensitive Snapshots in Wireless Sensor Networks. Technical Report OSU-CISRC-7/07-TR51, The Ohio State University, 2007.
8. V. Mittal, M. Demirbas, and A. Arora. Loci: Local clustering service for large scale wireless sensor networks. Technical Report OSU-CISRC-2/03-TR07, The Ohio State University, 2003.
9. S. Kulkarni and U. Arumugam. TDMA service for Sensor Networks. In *ICDCS*, volume 4, pages 604–609, 2004.
10. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Transacttions on Computer Systems*, 3(5):63–75, 1985.
11. S. Servetto. Sensing LENA - Massively Distributed Compression of Sensor Images. In *IEEE ICIP*, 2003.
12. F. Dehne, A. Ferreira, and A. Rau-Chaplin. Parallel fractional cascading on hypercube multiprocessors. In *Computational Geometry Theory Applications*, volume 2, pages 144–167, 1992.
13. J. Gao, L.J. Guibas, J. Hershberger, and L. Zhang. Fractionally cascaded information in a sensor network. In *IPSN*, pages 311–319, 2004.
14. Rik Sarkar, Xianjin Zhu, and Jie Gao. Hierarchical Spatial Gossip for MultiResolution Representations in Sensor Networks. In *IPSN*, pages 311–319, 2007.
15. G. Pei, M. Gerla, and T.-W. Chen. Fisheye State Routing in Mobile Adhoc Networks. In *ICDCS Workshop on Wireless Networks*, pages 71–78, 2000.
16. S. Kim. Sensor networks for structural health monitoring. Master's thesis, University of California, Berkeley, 2005.
17. V. Naik and A.Arora. Harvest: A reliable bulk data collection service for large scale wireless sensor networks. Technical Report OSU-CISRC-4/06-TR37, The Ohio State University, 2006.