Distance Sensitive Snapshots in Wireless Sensor Networks

Vinodkrishnan Kulathumani and Anish Arora
Deptartment 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. Applications often require that they be consistent and timely, which makes them 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.**

More specifically, we design generalized snapshot services for arbitrary sensor networks with N nodes that reside in an f-dimensional space, where each node periodically generates m bits of information. Our design is stepwise: we first design a service where a global snapshot S is periodically delivered to every node j in the network as fast as possible with *distance sensitive latency* such that the staleness of the state of a node i in S is $O(3^f * N * m * d)$ where $d = dist(i, j)$, but the communication cost averages to $O(N^2 * m)$ for N samples (one from each node). We refine the solution to improve the communication cost by adding the property of *distance sensitive resolution*, whereby the resolution of the state of i in S decreases as $O(d^f)$; the staleness of the state of i in S is now $O(3^{2f} * log(n) * m * d)$ and the communication cost is $O(3^f * log(n) * N * m)$, where $n = N^{\frac{1}{f}}$, for N samples. Next, we further refine the service to reduce the communication cost to either $O(3^f * N * (m + log(n/m)))$ or $O(3^f * N * m)$, by adding the property of *distance sensitive rate*, whereby the state of nearby nodes is updated more often than farther nodes. The memory required per node in our solutions is bounded by $3^f * log(n) * m$.

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, networks with arbitrary sized holes, networks with imperfect clustering and non unit disk radios. We also discuss how different control applications can exploit these generalized snapshots.

I. 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.

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 that that for nearer areas.

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.

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. Care has to be taken to ensure that information propagates in different directions at the same speed. 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. The aggregated data is then dispersed to all nodes at that level. This approach 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 is desired. To redress this problem, we disperse a summary computed at each level r not only to nodes in level r cluster but also to nodes in all neighboring level r clusters. Once again our scheduling and other protocol actions at each step are such that information at each level is aggregated and dispersed in a pipelined manner concurrently at all levels; by this, new information can be generated at a node as soon as previous information has been dispersed only to its local neighborhood.

> 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 $O(3^f * log(n) * N * m)$.

In the second algorithm, the snapshots of the network arrive at all nodes with the same rate. 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: $\forall a, b, S$ such that $S = a \cup b$, $f(S) = f(f(a) \cup f(b))$. Examples of such functions are average, max, min, count and wavelet functions.

We then relax our regularity assumptions and 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. When a grid location has more than 1 node, each node takes turns over multiple rounds to send its data resulting in time sharing between nodes to transmit their own data. Once data is sent out from the source, however, the forwarding of the data does not incur an extra delay despite going through denser grid locations. 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 locations. 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. When we relax the strictly 1 hop communication range to radio interference range varying from 1 to s hops, the basic scheduling for each round has to take into account this additional interference. This simply results in longer round lengths that are proportional to the size of interference region.

In our algorithms, we ensure in order arrival of state information from any given node. Therefore timestamps are not checked in the algorithms while updating the local data structure. As long as the application does not require timestamps associated with each state, global time synchronization may not be required. A local notion of time however is needed to ensure fair scheduling of transmission of nodes.

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 extend our algorithms to irregular networks. We discuss related work in Section 7 and make concluding remarks in Section 8.

II. 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. We let $j.X_i$ and $j.T_i$ (respectively, $j.X_p$ and $j.T_p$) refer to the state of node i (respectively, partition p) and the timestamp of that state at node j.

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. From Section 6 onwards, we remove each of these restrictive assumptions.

Generalized snapshots: Let's begin by considering global snapshots where state is maintained for each node.

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. More to the point, 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 recoding 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 and thus the decrease in resolution of the state of a node is bounded by the size of p. 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. In other words, j has a *myopic* or *fisheye* view of the network.

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.

Note that the concept of distance sensitive rate is orthogonal to that distance sensitive latency. In the latter staleness in the state received decreases with distance but fresh information arrives at the same rate at all nodes, where as in the former, the state of nearby nodes is reported more often than farther nodes.

III. Distance sensitive latency snapshots

In this section, we first describe a simple algorithm that has distance sensitive latency albeit the latency is non-uniform across different directions. We then refine the algorithm by introducing a single level of clustering so as to achieve uniform information flow.

A. Snapshots with non-uniform latency

Schedule: We schedule nodes to transmit maximally while avoiding interference. In our assumed grid topology communications model, nodes separated by 3 units horizontally or vertically can transmit simultaneously without mutual interference. Each node gets a turn to transmit once every 9 slots. The required duration of a slot is a critical part of the algorithm and is derived in the following analysis.

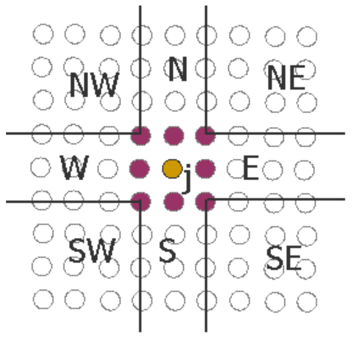

Fig. 1. Zones around a node j

Algorithm $S0$**:** The algorithm has 3 actions at each node i. Whenever it is is turn to transmit, i first updates its local data structure with its own state and transmits a message containing the state of all nodes, as maintained in its local data structure. When i hears a message from any of its 9 neighbors, it applies the following update rules. The network around i is divided into 8 zones as shown in the Fig. 1. When i receives a message from (say) its north neighbor, the state of the nodes in zone n are updated, and so on. Updating thus yields the following property.

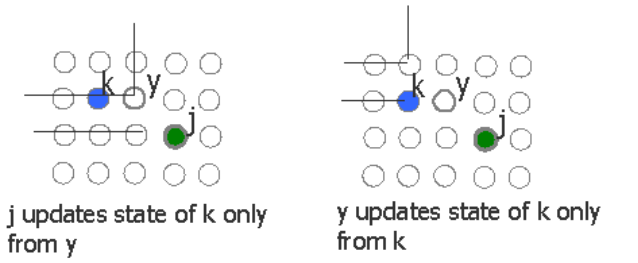
k
y
j
j updates state of k only from y
k
y
j
y updates state of k only from k

Fig. 2. Unique path $k-y-j$ from node k to node j

Lemma III.1. *In* $S0$, *state of any node* i *is transmitted to some other node* j *along exactly one path in the grid at all times.* □

The lemma is illustrated in Fig. 2. $S0$ thus maintains a unique channel for communicating from node i to node j. This ensures that state of i in the local data structure of j is not over-written by stale data from another path. We now analyze the latency and communication cost of scheme $S0$.

Lemma III.2. *In* $S0$, *the minimum slot width per node so that global snapshots can be delivered periodically at all nodes is* $\frac{N*m}{W}$.

Proof: During each slot, a node transmits its local data structure which is $N*m$ bits. The rate of transmission is W bits per second. The required slot width (s_w) is therefore $\frac{N*m}{W}$. ■

Lemma III.3. *In* $S0$, *the maximum staleness in the state of a node* i *received by a snapshot at node* j *is* $O(N*m*d)$ *where* $d = dist(i,j)$.

Proof: Node i updates its state just before transmitting. From then on, each successive recipient can forward this information in at most 8 slots per hop. Thus, j learns about the state of i in time at most $(8*(dist(i,j)-1))*s_w$, where $dist(i,j)$ is the hop distance between i and j. ■

In $S0$, although snapshots are delivered with distance sensitive latency, the latency is non-uniform across different directions. In the grid topology, the latency varies between $O(d)$ and $O(8*d)$. Uniformity is a desirable property especially when aggregation needs to be performed. We now describe a refinement of $S0$ that has uniform distance sensitive latency.

B. Snapshots with uniform latency

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. 3.

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 contstraints is one where all level 0 nodes take turns in say a clockwise direction. For example, level 0 nodes transmit in a clockwise direction starting from $j.ne$, where j is a level 1 node. Each round thus consists of 17 slots.

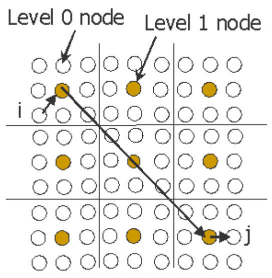

Fig. 3. 1 level clustering

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. 4.

- 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. Information flow from i to j is illustrated in Fig. 3. 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.

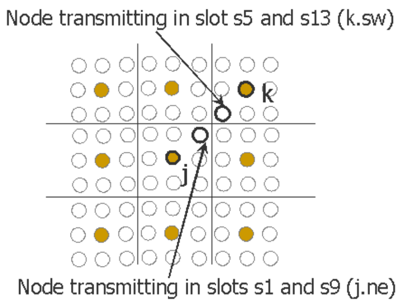
Node transmitting in slot s5 and s13 (k.sw)
k
j
Node transmitting in slots s1 and s9 (j.ne)

Fig. 4. Neighboring level 1 clusters j and k

Lemma III.4. *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 III.5. *In $S0$ and $S1$, the average communication cost to deliver a global snapshot to all nodes per sample from each node is $O(N^2 * m)$.*

Proof: To deliver a snapshot corresponding to 1 sample from each node, every node communicates $O(m*N)$ bits n times. And to deliver a snapshot corresponding to y samples from each node, every node communicates $O((n + y) * (m * N))$ 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 one sample from each node to all nodes is $O(m * N)$. The average communication cost over N nodes is $O(N^2 * (m)$. ■

Extending to other dimensions: Consider an f dimensional structure. In this 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. The number of slots per round increases proportional to 3^f. Using these we get the following lemma.

Lemma III.6. *In $S1$, the maximum staleness in the state of i in a snapshot S received by j in a network of f dimensions is $O(3^f * N * m * d)$ where $d = dist(i,j)$.* □

IV. 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. 5. 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. 5, 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. This algorithm can be implemented in multiple phases; aggregates at each level are computed and dispersed sequentially. When implemented sequentially, however, new samples can be generated at each node only after data is dispersed at all levels. To avoid this constraint, we use the following pipelined implementation described next.

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.

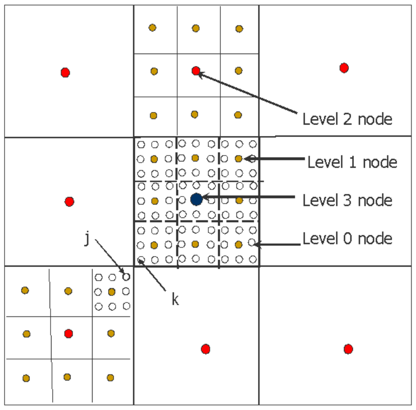

Fig. 5. Hierarchical clustering

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. 6, 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: We continue to schedule the nodes to transmit in rounds as we did in algorithm $S1$. In the first slot, all level 1 clusterheads transmit. In the remaining slots, all the level 0 nodes per cluster take turns and transmit twice.

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.

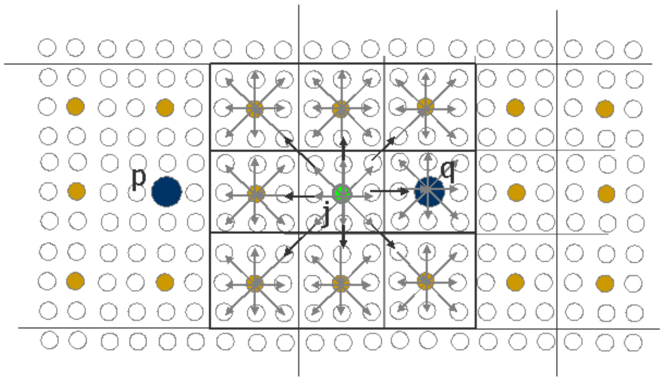

Fig. 6. Illustrating level 1 tree rooted at j

Algorithm $S2$**:** We describe the actions executed by the nodes in three parts: (1) send / compute actions for nodes with $j.L > 0$, (2) send / compute actions for nodes with $j.L = 0$ and (3) receive / update actions for all nodes.

- In slot 0 of each round nodes with $j.L > 0$ compute the summary $M(r,j)$ for each level $1 \le r \le j.L$ 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 $1 \le r \le j.L$.
- 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. 4.
  - In slot 1, 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 slot 9, 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. 6, 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. We now analyze the latency and communication cost of algorithm $S2$.

Lemma IV.1. *In $S2$, the maximum message length needed per slot is $(9 * log(n) - 7) * m$ bits.*

Proof: Consider a node j at any level r, where $0 \leq r \leq log(n)$. Let $1 \leq l \leq log(n) - 1$. A level l tree rooted in the same level l cluster as that of j can pass through j. A level l tree rooted in neighboring level l clusters as that of j can also pass through j, but no other level l tree can pass through j. Thus, 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. The maximum message length needed per slot in algorithm $S3$ is $(9 * log(n) - 7) * m$ bits. ■

It follows that the slot width s_w needed in algorithm $S3$ is $\frac{(9*log(n)-7)*m}{W}$ bits per second.

Lemma IV.2. *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 $p.nbr(r)$. Thus, a level r summary is computed based on a level $r-1$ summary that was generated $(17/3) * 3^{r-1} * s_w$ time ago, since latency between each pair of level 1 nodes is 17 slots. A level $r-1$ summary is computed based a level $r-2$ summary, and so on until level 0. Upon summation, we see that 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 $(3/2) * 3^r$. The maximum latency involved is $(17/2) * 3^r * s_w$. 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 maximum staleness in the state of a node i at node j is given by the following equation:

$$\begin{aligned} staleness(i,j) &= (L1 + L2) && (1)\\ &= \frac{(L1+L2)}{3^{r-1}} \times 3^{r-1} && (2)\\ &= O(34 * s_w * d) && (3)\\ &= O(log(n) * m * d) && (4) \end{aligned}$$

The result follows.

■

Lemma IV.3. *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 IV.4. *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 $O(N * (m * log(n))$.

Note that if y is small, for instance, if there is only one sample from each node, then the communication cost is $O(N * n * (m + log(n/m))$. Pipelining the delivery of snapshots improves the average communication cost to $O(N * (m * log(n))$. ■

Lemma IV.5. *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 where $n = N^{\frac{1}{f}}$, 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. Also, as described in Section 3, the length of each round is $O(3^f)$. Using these, we generalize Lemmas 4.2 and 4.4 as follows:

Lemma IV.6. *In $S2$, the maximum staleness in the state of a node i received by a snapshot at node j in a network of f dimensions is $O(3^{2f} * log(n) * m * d)$ where $d = dist(i, j)$.* □

Lemma IV.7. *In $S2$, the average communication cost in the network to deliver a snapshot of one sample from each node to all nodes is $O(N * 3^f * log(n) * m)$.* □

Note also that in an f dimensional structure, the state of 3^{f*r} nodes is compressed in m bits. Following the structure of proof for lemma $IV.3$, we get the following result.

Lemma IV.8. *In $S2$, the resolution of state of a node i in a snapshot received at node j in an f dimensional network is $\Omega(\frac{1}{d^f})$ where $d = dist(i, j)$.* □

V. 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.

A. 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 $S2$. The snapshot at each level is aggregated into m bits. 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 $log(n)$. The actions executed by every node in their corresponding slots remain the same as in algorithm $S2$ except that every level r summary is now transmitted over $min(3^r, m)$ rounds, as opposed to in each round containing a new level r summary.

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 analyze the latency and communication cost of algorithm $S3a$.

Lemma V.1. *In $S3a$, the maximum message length needed per slot in algorithm is $11 * \frac{m}{2} + 9 * log(\frac{n}{m})$ bits.*

Proof: Recall that at most 9 trees at levels $1..log(n)-1$ can pass through each node, and there is only one level $logn$ tree. Also, j belongs to only one level 0 tree for which it is not a leaf. Moreover, $\frac{m}{3^r}$ bits are allocated for each level $0 \leq r \leq log(m)$ and 1 bit for each level z where $log(m) < z \leq log(n)$. The maximum message length (ML) needed is obtained by the following equation:

$$ML = \frac{m}{3^0} + \sum_{i=1}^{i=log(m)} (9 * \frac{m}{3^i}) + 9 * (log(n/m)) \quad (5)$$

$$= m + m * \frac{9}{2} * (1 - \frac{1}{m}) + 9 * log(\frac{n}{m}) \quad (6)$$

$$< m + m * \frac{9}{2} + 9 * log(\frac{n}{m}) \quad (7)$$

The result follows from these facts. ■

It follows that the slot width s_w needed in algorithm $S3a$ is $\frac{m * \frac{11}{2} + 9 * log(\frac{n}{m})}{W}$ bits per second.

Lemma V.2. *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 $d = dist(i, j)$.*

Proof: Consider a node p at level r where $r \leq log(m)$. 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 $p.nbr(r)$. The latency between each pair of level 1 nodes is 17 slots. Thus the latency to travel from level $r-1$ node to level r node is given by $(17/3) * 3^{r-1} * s_w$. But in this algorithm, a level $r-1$ summary is actually transmitted in $3^(r-1)$ rounds by dividing it into 3^{r-1} parts. Thus, a level r summary is computed based on level $r-1$ summary that was generated $2 * (17/3) * 3^{r-1} * s_w$ time ago. A level $r-1$ summary is computed based a level $r-2$ summary and so on until level 0. Upon summation, we see that the staleness of a level 0 state information in a level r summary is $(17) * 3^{r-1} * s_w$.

Note that a complete level $r-1$ snapshot is sent every 3^{r-1} rounds in a pipelined manner. Thus every 3^{r-1} rounds, a level $r-1$ snapshot is received by a node. On the other hand a level r snapshot is computed only every 3^r rounds. Thus a fresher level $r-1$ snapshot is always available to compute a new level r snapshot.

The maximum distance traveled by a level r summary is $(3/2) * 3^r$. However, this summary is sent in 3^r rounds. The maximum latency involved is $2 * (17/2) * 3^r * s_w$.

Recall that in order to update the local data structure of j, the state of a node i is updated using summary $M(x', y')$ where x' is the lowest level which contains information about k. Now 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 maximum staleness in the state of a node i at node

j is given by the following equation:

$$\begin{aligned} staleness(i,j) &= (L1+L2) && (8)\\ &= \frac{(L1+L2)}{3^{r-1}} \times 3^{r-1} && (9)\\ &= O(68 * s_w * d) && (10)\\ &= O(m*d+log(n/m)*d) && (11)\end{aligned}$$

Note that at levels $r > log(m)$, 1 bit is allocated per round. Thus, for all these levels, a summary can be sent out in less than 3^r rounds. The maximum staleness for those nodes whose state is obtained using summaries greater than level r is less than that derived in the above equation.

The maximum staleness in the state of a node i received by a snapshot at node j is thus $O((m+log(n/m))*d)$, where $d = dist(i,j)$. ■

Lemma V.3. *In $S3a$, the maximum interval between when a node j receives the state of node i is $O((m + log(n/m)*d)$, where $d = dist(i,j)$.*

Proof: Consider levels $0 \le r \le log(m)$. Note that a complete level r snapshot is sent every 3^r rounds in a pipelined manner. Thus every 3^r rounds, a level r snapshot is received by a node. The time corresponding to 3^r rounds is $(17/3)*3^r*s_w$.

Recall that in order to update the local data structure of j, the state of a node i is updated using summary $M(x',y')$ where x' is the lowest level which contains information about k. Now 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 maximum interval between when a node j receives the state of node i is given by the following equation:

$$\begin{aligned} interval(i,j) &= \frac{(17/3)*3^r*s_w}{3^{r-1}} \times 3^{r-1} && (12)\\ &= O(17*s_w*d) && (13)\\ &= O(m*d+log(n/m)*d) && (14)\end{aligned}$$

Note that for levels $r > log(m)$, 1 bit is allocated per round. Thus, for all these levels, a summary can be sent out in less than 3^r rounds. The maximum interval between receiving two successive state information for those nodes whose state is obtained using summaries greater than level r is less than that derived in the above equation. The maximum interval between when a j receives the state of i is thus $O((m+log(n/m)*d)$. ■

Lemma V.4. *In $S3a$, the average communication cost to deliver a snapshot of one sample from each node to all nodes is $O(N*(m+log(n/m))$.* □

Proof: To deliver a snapshot corresponding to 1 sample from each node, every node communicates $O(m+log(n/m))$ bits n times. And to deliver a snapshot corresponding to y samples from each node, every node communicates $O((n+y)*(m+log(n/m)))$ 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 one sample from each node to all nodes is $O(m+log(n/m))$. The average communication cost over N nodes is $O(N*(m+log(n/m)))$. ■

Discussion: We note that apart from decreasing the average communication cost per sample from each node, solution $S3a$ also offers flexibility in terms of the size of each sample. When $m << log(n)$, the factor $log(n)$ dominates and the average communication cost is $O(N*log(n))$ and the staleness is bounded by $O(log(n)*d)$. In algorithm $S2$ the corresponding costs are $O(N*m*log(n))$ and $O(m*log(n)*d)$ respectively. But when the sample size increases to as large as order n, the average communication cost is $O(N*n)$ and the staleness is bounded by $O(n*d)$, where as in algorithm $S2$ the corresponding costs are $O(N*n*log(n))$ and $O(n*log(n)*d)$ respectively.

B. Distance sensitive rate by time division

Again, we hierarchically partition the network into clusters and schedule nodes to transmit in rounds exactly as we did in algorithm $S3$. The snapshot at each level is aggregated into m bits. 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 in that round. The frequency at which a round is allocated to a particular level increases exponentially as level decreases. Every alternate round is allocated to level 1. Once every 4 rounds is allocated to level 2 and so on. This results in a level k aggregate being sent out every 2^k rounds.

Algorithm $S3b$: Consider level $r > 0$. Let the rounds be numbered starting from 1. The first round that is allocated to level r is 2^{r-1}. Starting from that round, all successive rounds 2^r apart are allocated to level r. Thus, all rounds enumerated by $2^{r-1} + i \times 2^r$ for $i > 0$ are allocated to round r. A level 0 information is carried in all rounds.

We refine algorithm $S2$ as follows. Consider a round s that belongs to level $r_s > 0$. A node j with level $j.L \ge r_s$ 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. We now analyze the latency and communication cost of algorithm $S3b$.

Lemma V.5. *In $S3b$, the maximum message length needed per slot is $10*m$ bits.*

Proof: Recall that at most 9 trees at levels $1..log(n)-1$ pass through each node. But, in each round, information pertaining to only one of those levels is forwarded resulting a message size of $9*m$ bits. Also, j belongs to only one level 0 tree for which it is not a leaf, resulting in m bits for level 0 in each round. The result follows. ■

We now state the following lemmas regarding latency, rate of snapshot delivery, and communication cost of algorithm $S3b$.

Lemma V.6. *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 $d = dist(i,j)$.* □

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 $p.nbr(r)$. The latency between a pair of level 1 nodes is 17 slots. However, between every pair of level 1 nodes a level $r-1$ information is forwarded only once every 2^{r-1} rounds. Therefore delay over 3^{r-1} distance is $(17/3)*3^{r-1}*2^{r-1}*s_w$. Thus, a level r summary is computed based on level $r-1$ summary that was generated $(17/3)*3^{r-1}*2^{r-1}*s_w$ time ago. A level $r-1$ summary is computed based a level $r-2$ summary and so on until level 0. Thus the maximum staleness ($L1$)of a level 0 state information in a level r summary is $(17/2)*3^{r-1}*2^{r-1}*s_w$.
The maximum distance traveled by a level r summary is $(3/2)*3^r$. The maximum latency ($L2$) involved is $(17/2)*3^r*2^r*s_w$.
Recall that in order to update the local data structure of j, the state of a node i is updated using summary $M(x',y')$ where x' is the lowest level which contains information about k. Now 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 maximum staleness in the state of a node i at node j is given by the following equation:

$$\begin{aligned}
staleness(i,j) &= (L1+L2) && (15)\\
&= \frac{(L1+L2)}{3^{r-1}} \times 3^{r-1} && (16)\\
&= \frac{s_w*(119/2)*6^{r-1}}{3^{r-1}} \times 3^{r-1} && (17)\\
&= s_w*(119/2)*2^{r-1}*d && (18)\\
&= O(s_w*d^2) && (19)\\
&= O(m*d^2) && (20)
\end{aligned}$$

Thus the maximum staleness in the state of a node i received by a snapshot at node j is $m*d^2)$ where $d = dist(i,j)$. ■

Lemma V.7. *In $S3b$, the maximum interval between when a node j receives the state of node i is $O(m*d)$.* □

Proof: Note that a level r information is forwarded once every 2^r rounds. Thus when snapshots are delivered in a pipelined manner, a new level r snapshot is received at every node once every 2^r rounds. The latency of each round is 17 slots. Time corresponding to 2^r rounds is $(17/3)*2^r*s_w$
Recall that in order to update the local data structure of j, the state of a node i is updated using summary $M(x',y')$ where x' is the lowest level which contains information about k. Now 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 maximum interval between when a node j receives the state of node i is given by the following equation:

$$\begin{aligned}
interval(i,j) &= \frac{(17/3)*2^r*s_w}{3^{r-1}} \times 3^{r-1} && (21)\\
&= O(s_w*d) && (22)\\
&= O(m*d) && (23)
\end{aligned}$$

■

Lemma V.8. *In $S3b$, the average communication cost to deliver a snapshot of one sample from each node to all nodes is $O(N*m)$.*

Proof: To deliver a snapshot corresponding to 1 sample from each node, every node communicates $O(m)$ bits n times. And to deliver a snapshot corresponding to y samples from each node, every node communicates $O((n+y)*(m)$ bits since all the y samples are delivered in a pipelined manner. Hence if y is large and $y = \Omega(n)$, the average communication cost at each node to deliver a snapshot of one sample from each node to all nodes is $O(m)$. The average communication cost over N nodes is $O(N*m)$. ■

Lemma V.9. *In $S3a$ and $S3b$, the memory requirement per node is $O(log(n)*m)$.* □

Proof: The maximum number of trees passing through any node is $O(log(n))$. The storage at each node is the most recent $M(x,y)$ for each $tree(x,y)$ that a node belongs to. In algorithm $S3a$, the difference is that this information ($(M(x,y))$ arrives at a node over multiple rounds. The memory requirement is still $O(log(n)*m)$. ■

Both algorithms $S3a$ and $S3b$ can be generalized to f dimensions just as algorithm $S2$ is [7]. We summarize our results in Fig. 7, which shows the bounds on staleness and resolution in the state of i at nodes that are distance d away, the bound on interval at which state updates of i are received, and the average communication cost for delivering a snapshot of N samples (one from each node) across the network.

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. 7. Summary of results for snapshot algorithms

VI. Irregular networks

In this section, we relax several assumptions weve made in designing algorithms $S0 - S3$. We show how the algorithms continue to yield distance sensitive snapshots and quantify the impact on the performance in the following cases: non uniform density, holes of arbitrary sizes within the connected network, non unit disk radios and imperfect clustering.

Clustering Model: 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 may not assume, for example, that all level 0 nodes are within 1 hop range of the level 1 clusterhead. 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. Moreover, the paths between neighboring clusters also need not the shortest ones, unlike what we assumed in the previous sections.

More formally, our clustering assumption is stated as follows (with distance standing for communication *hop* distances):

- ($C1$) All nodes within distance $\frac{3^k-1}{2}$ from a level k clusterhead belong to that cluster.
- ($C2$) The maximum 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.

A. Networks with non uniform density

Once the network has been partitioned into clusters, we impose a virtual grid on the network, as shown in Fig. 8. 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. We first describe how the case of over density is handled.

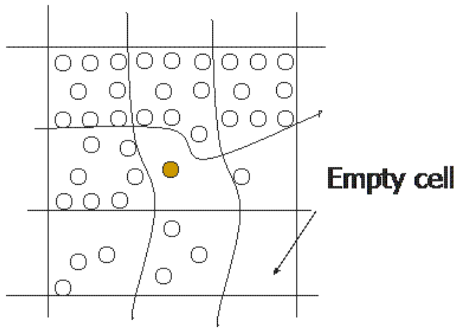

Fig. 8. Virtual grid and cells with different densities

Over density cells: In the virtual grid, each cell gets a slot to transmit as described in algorithms $S0-S1$. 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: If a particular cell is empty in sparse regions of the network, then the communication path between neighboring level k clusters ($0 < k < log(n)$) is lost. We consider 2 density models for network with holes. In the first model, the density of the network is such that the communication hop distances are of the same order as the geometric distance despite holes in the network. We call this density assumption, DA. In the second model, DA does not hold, i.e., the network may have arbitrary sized holes. The hop distance between any two nodes may be arbitrarily higher than the Euclidean distance. For both these density models, 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 2 is adapted as follows. For ease of explanation, assume $z = 2$. Thus, a level 1 cluster comprises a minimum of the 1 hop neighborhood and a maximum of the 2 hop neighborhood. At the beginning of each round, level 1 clusterheads transmit.

There can be at most 26 level 0 nodes in each cluster. Some may be absent because of holes or because of non uniform clustering. Depending on the position in the cluster, each level 0 node gets 2 slots per round such that information is exchanged from all 8 directions. The round length thus increases with z.

Distance sensitivity: A level k tree rooted at any node j spans all nodes in its own level k cluster and all nodes in the neighboring level k clusters, using fixed paths as described in Sections 3 and 4. If a neighboring level k cluster is completely absent, then there is no need to reroute the information. However if the neighboring level k cluster is present but the path is broken, then information has to be re-routed. We now investigate the impact on latency and resolution when the information takes a different path than normal.

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 p. We consider DA networks and non DA networks seperately.

DA **networks:**

Lemma VI.1. *For DA networks, 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 VI.2. *For DA networks, algorithms $S1$, $S2$, $S3a$ 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}. Using $C3$ and $C4$ we have that level k information can be exchanged between i and j.

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 $S1$, $S2$, $S3a$ and $S3b$ are sufficient to transmit all information despite the trees not being created in a regular pattern. ■

Rerouting in Case 1 is illustrated in Fig. 9. 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 DA **networks:** In these networks, Lemma $VI.1$ may not hold. The clustering may be such that distance between two nodes may be small, but they may belong to neighboring clusters at a much larger level. One such instance is illustrated in Fig. 10 where there are two dense regions $D1$ and $D2$ connected by a thin strip and a vertical thin section of nodes. In this instance, $dist(A, B) = 4$ units. R and S are level $k'-1$ clusterheads that contain A and B respectively. P and Q are level k' clusterheads that contain A and B respectively. The clustering is such that A and B are not neighbors up till an arbitrarily high level k' because the neighboring clusterheads for all levels smaller than k' lie in the thin long section of the network. In such cases, if a level k information is transmitted only up to neighboring level k clusters, distance sensitivity will not be satisfied. In Fig. 10, nodes A and B will exchange information only at level k'. We therefore refine our algorithms as follows.

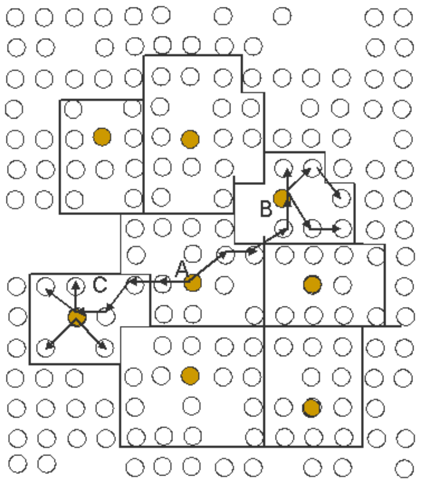

Fig. 9. Handling holes in dense networks

Refinement R**:** Instead of extending a level k tree only up to all nodes in neighboring level k clusters, we extend a level k tree until the height of the tree is equal to $3 * z^k * \frac{(3^k-1)}{2}$ along all branches. But we retain that the degree of each node is still bounded by 8. □

In other words, we extend a tree up to a height equal to the maximum radius of a cluster plus the maximum diameter of a cluster. This results in level k information propagating up to a hop distance that is proportional to 3^k regardless of whether all nodes in that radius are limited to neighboring clusters; this is stated formally in the following lemma.

Lemma VI.3. *If k is the smallest level of information received by a node j about the state of i, then $p \geq z^{k-1} * 3^{k-1}$.*

Proof: Node j does not receive level $k-1$ information about i. Therefore, the level distance of j from level $k-1$ clusterhead of i is greater than $3 * z^{k-1} * \frac{(3^{k-1}-1)}{2}$. The maximum distance of i from its level $k-1$ clusterhead is $z^{k-1} * \frac{(3^{k-1}-1)}{2}$. Thus, $p \geq z^{k-1} * 3^{k-1}$. ■

Theorem VI.4. *For non DA networks, algorithms $S1$,*

$S2$, $S3a$ and $S3b$ with refinement R yield snapshots that retain their distance sensitive properties.

Note: Refining the tree to go beyond neighboring clusters may violate the capacity bound at each node per slot. This is because certain nodes may have more than 9 trees passing through them at any level as trees are not limited to those from neighboring clusters any more. Along links that have more trees passing through them, information propagation has to be time shared. This results in higher latency when passing through such parts of the network. Consider Fig. 10 again. In this graph, a level $k' - 1$ tree rooted inside R will extend not only to neighboring $k' - 1$ cluster T, but also to nodes inside S. Since S is a dense region, nodes within S may already have 9 level $k' - 1$ trees passing through them and the tree from R is an addition. This results in information flow becoming slower when passing through nodes in S.

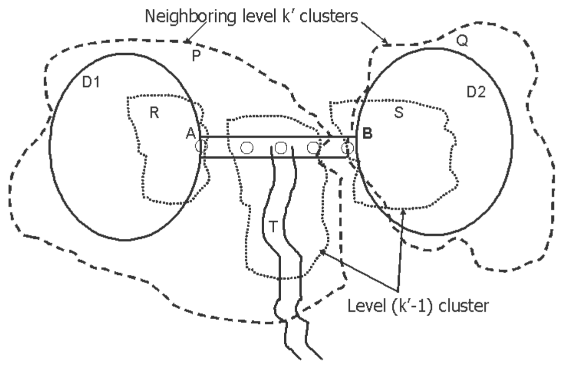

Fig. 10. Handling holes in sparse networks

B. Non uniform radio range

The design of algorithms $S0 - S3$ assumed a strictly 1 hop communication 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.

C. Implementation considerations

In this subsection, we highlight considerations for implementing our snapshot services in wireless sensor networks. In the past, we have implemented *Trail* [5], an asynchronous network protocol for querying object states and used this in the context of a distributed object tracking application. We have noted that as the objects in the network get densely located, interference issues lead to high latency and decreased efficiency. The snapshot services that we consider in this paper are high density operations and TDMA 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. Another issue to consider is that of localization. For our snapshot services, knowledge of relative location with respect to clusterheads is sufficient as opposed to the knowledge of precise coordinates.

VII. Related work

Communicating periodic global state snapshots is a well studied problem in distributed systems [9] 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 [10], the authors propose a framework for a one time all-to-all broadcast of sensor data assuming the data is spatially correlated. There has also been work [11] on compression mechanisms for correlated sensor data sent to a central base station. Instead, in this paper we explore an alternate form of compression to achieve efficiency by generalizing the notion of snapshots to satisfy certain distance sensitive properties. 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. Each node stores an ordered list of keys, shares a well distributed sample of that list with its neighbors, a sample of that sample with its two hop neighbors, and so on. 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. Stored data 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. Accordingly we describe push based services that regularly deliver to subscribers snapshots of the network in a pipelined manner. By providing snapshots with not just distance sensitive -resolution but also -latency and -rate, we achieve compression and thereby efficiency. At the same time these services can be used in range based querying as well as in several other control applications.

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. The spatial gossip protocol [16] is one in which each node in a peer-to-peer network chooses to communicate to another node with a probability that decreases polynomially with the distance between the pair.

In [17], the authors propose a probabilistic algorithm to aggregate sparse data generated in sensor networks that is eventually exfiltrated from the network. In contrast, we have focused on dense data generation models. Synchronous, push based services are more suited in this model as they are less prone to contention in sensor networks.

Recently algorithms for bulk data collection in sensor networks have been proposed. In [18] data is collected from one node at a time, while [19] performs concurrent, pipelined exfiltration of data using TDMA schedules. In [20], the authors describe TDMA based algorithms optimized for convergecast. Our algorithms can be specialized for the case of bulk convergecast and we additionally emphasize on efficiency using distance sensitive properties.

VIII. Conclusion

We have generalized the basic notion of snapshots using distance sensitive notions and accordingly designed efficient wireless sensor network algorithms that periodically deliver. Our algorithms can be formulated to allow delivery at a subset of nodes as opposed to all nodes. They are memory efficient and realizable in networks with irregular density, with arbitrary sized holes, imperfect clustering, and non unit disk radios. We have quantified the maximum rate at which information can be generated at each node so that snapshots are periodically delivered across the network. We have specified the allowable aggregation functions in abstract terms, allowable functions include average, max, min and wavelet functions.

Our algorithms assume knowledge of location at each node. We map nodes to a virtual grid and each node is aware of the clusters it belongs to and position within each cluster. We know that algorithms for distance sensitive snapshots can be designed without knowledge of location at each node, although it is open as to what are the weakest assumptions under which the problem is solvable.

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. Thus far, we have only a preliminary version of our distance sensitive latency snapshot service for linear networks [21], and this implementation has been used to support a pursuer evader tracking application that was demonstrated in Richmond Field Station in August 2005.

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] M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM Transacttions on Computer Systems*, 3(5):63–75, 1985.

[10] S. Servetto. Sensing LENA - Massively Distributed Compression of Sensor Images. In *IEEE ICIP*, 2003.

[11] T. Dang, N. Bulusu, and W. Feng. Robust Information Driven Data Compression Architecture for Irregular WSN. In *EWSN*, 2007.

[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] D. Kempe, J. M. Kleinberg, and A. J. Demers. Spatial gossip and resource location protocols. In *ACM Symposium on Theory of Computing*, pages 163–172, 2001.

[17] J. gao, l. Guibas, N. Milosavljivec, and J. Hershberger. Sparse Data Aggregation in Sensor Networks. In *IPSN*, pages 430–439, 2007.

[18] S. Kim. Sensor networks for structural health monitoring. Master's thesis, University of California, Berkeley, 2005.

[19] 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.

[20] S. Kulkarni and U. Arumugam. TDMA service for Sensor Networks. In *ICDCS*, volume 4, pages 604–609, 2004.

[21] V. Kulathumani, A. Arora, M. Sridharan, and M. Demirbas. Trunk and Trail: WSN Services for Distributed Object Tracking. Technical Report OSU-CISRC-11/05-TR72, The Ohio State University, 2005.