A Case For Resource Heterogeneity in Large Sensor Networks

Srikanth Kandula
CSAIL
MIT, Cambridge MA 02139

Jennifer Hou, Lui Sha
Dept. of Computer Science
UIUC, Urbana IL 61801

ABSTRACT

Sensor networks have traditionally consisted of nodes with the same amount of resources such as battery life and computational power. While such homogeneity has distinct advantages in terms of ease-of-fabrication, it has also been shown that in multi-hop ad-hoc scenarios homogeneity leads to large duty cycles, small end-to-end data throughput and poor deployment lifetimes. In this paper, we investigate sensor networks that have a single degree of heterogeneity, a random subset of the sensors, called accumulators, have more power and computational capability.

To this end, we develop a decentralized, hierarchical clustering algorithm, called *Hierarchical Clustering and Routing (HCR)* algorithm. HCR exploits heterogeneity among sensor nodes to form a cluster hierarchy, with the objective of achieving better information throughput and improving network lifetime. A unique feature of HCR is its integration of routing with cluster formation and data delivery. Routing tables are constructed/updated in the course of cluster formation and data transmission, therefore incurring essentially no routing overhead.

By exploiting geometrical features of hexagons, we show several desirable properties of HCR. In particular, we show via *ns-2* simulations that HCR improves message overhead in cluster formation by 120-210%. We also show that heterogeneity achieves performance improvements of (i) upto 200% in terms of information throughput, (ii) power savings of upto 30% of the power spent in data transmission and (iii) a 2% density of accumulators is sufficient for most improvements.

1. INTRODUCTION

Recent technological advances have led to the emergence of small, low-power devices that integrate sensors and actuators with limited on-board processing and wireless communication capabilities. Pervasive networks of such sensors and actuators open new vistas for constructing complex control systems. Unlike traditional wired or wireless networks, sensor networks are application-specific and data-centric. Knowledge of data traffic patterns, such as one or more sources to one or more sinks, can be used to design infrastructures that are better suited to the needs of the application. Examples of such networks include, audio-visual surveillance systems consisting of autonomous motion detectors and acoustic monitors for homeland security, airborne deployment of similar devices to establish military surveillance networks and temperature/pressure/humidity measuring devices for environmental and home monitoring.

Sensor deployments have traditionally consisted of large wireless, multi-hop ad-hoc networks of nodes that have limited battery life and on-board processing capabilities. While such resource constraints have been motivated by size and cost considerations, economies of scale and ease-of-fabrication, have led to homogeneity among sensor nodes, i.e. all sensors have very similar resource capabilities. At the same time, as more than one sensor may observe the same phenomenon, the information collected by sensors may be correlated, redundant, and/or of different qualities. Since each sensor is also responsible for relaying downstream sensor data to the sink, transmission of large volumes of redundant, low-quality information is an un-necessary drain on battery power and network capacity. The capability to process sensor data close to the point of its origin could therefore significantly improve the quality of network traffic (and a reduction in volume) leading to improved deployment lifetimes and reduced congestion losses (better network goodput). Such close-to-origin data processing capability could also serve several application-specific purposes, each of which are benefited by closer proximity to data, such as triangulation of intruder locations from sensor input quickly, generation of message digests to be transported to the sink, low-latency measurement based feedback control of sensor location, sensor calibration and maintenance of a distributed repository of sensor information that can be queried-when-needed. Advanced data processing, however, would require much higher processing and power capabilities than are traditionally present in modern sensors.

In this paper, we investigate sensor networks that have a single degree of heterogeneity. A randomly chosen subset of the sensors, called accumulators, are equipped with greater computational capabilities and battery life, while the vast majority of the network consists of cheaper, low-resource sensors. These accumulators provide data processing capabilities much closer to the point-of-origin. Such sensor networks present a key technical challenge in cooperative engagement — how to effectively coordinate and control sensors, through a random deployment of accumulators, in order to improve both network lifetime and goodput. We tackle this challenge by devising an effective algorithm for randomly distributed sensors and accumulators to self-organize themselves into hierarchical structures that efficiently utilize the available processing capabilities.

We develop, and rigorously prove the optimality of a decentralized, hierarchical clustering algorithm, called *Hierarchical Clustering and Routing (HCR)* algorithm. HCR exploits heterogeneity among sensor nodes to form a cluster hierarchy, with the objective of achieving better information throughput and improving network lifetime. Specifically, we envision a two-level, hierarchical cluster structure in which a level-0 cluster is statically composed of sensors in a hexagon in a two-dimensional hexagonal grid, and a level-1 cluster is “centered” at an energy-rich sensor device, called an *accu-*

mulator. A level-1 cluster is dynamically formed by all the level-0 hexagons (and the sensors therein) for whom the accumulator is the closest accumulator in the vicinity. Sensors gather data and send it, either periodically or on-demand to their respective level-0 cluster heads which then relay to the level-1 cluster head, the nearest accumulator. The accumulator is responsible for generating a single, concise digest from all the sensor data (and other data processing duties), and sending the digest to a data sink. Raw data, that is redundant and probably of poor quality is therefore transmitted only from the originating sensor up to the closest accumulator. Traffic on the long-haul toward the sinks, consists entirely of processed data. This leads to mitigation of message contention and network congestion, thereby leading to better network goodput. Network life-time is improved due to two factors. First, route redundancy in level-0 clusters allows idle sensors to "turn-off" radios without impairing network connectivity. Second, the amount of long-haul traffic that each node has to relay significantly decreases, leading to a reduction in power spent in relaying cross-traffic. Another unique feature of HCR (in contrast to other existing clustering algorithms that usually separate routing from cluster management) is that it constructs/updates routing tables in the course of cluster formation and thus incurs essentially no routing overhead.

By exploiting geometrical properties of hexagons, we show (i) the number of level-0 clusters whose transmission activities may interfere with that of a given level-0 cluster is smaller under the hexagonal cluster structure than under the square cluster structure; (ii) the control overhead incurred in HCR is the least possible, (worst-case $O(N)$, where N is the number of live hexagonal level-0 clusters) for any similar cluster formation mechanism and (iii) the network connectivity is preserved after clustering except under rare conditions. All the desirable properties are corroborated by *ns-2* simulation. In particular, we show via *ns-2* simulation that HCR achieves performance improvements of (i) 120%–210% in terms of message overhead in cluster formation over a naive algorithm (of message overhead worst-case $O(N^2)$, where N is the number of level-0 clusters), (ii) 200% in terms of information throughput over the baseline case of no hierarchy, (iii) up to 30% lower power consumption in relaying packets and (iv) significant savings of power incurred in the "idle" state.

Several clustering algorithms have been proposed in the literature, such as the k-cluster-based routing scheme [7], the zone routing protocol (ZRP) [4], the spine routing framework [3], and the min ID/max degree scheme [6]. However, to the best of our knowledge, HCR is the first such algorithm to be specifically targeted for data-centric sensor networks with sensor heterogeneity. Local power conservation techniques have been an active area of research [8, 2]. In contrast, the novel contribution of HCR is its emphasis on reducing power consumption in relay nodes as opposed to idle nodes. Finally, data dissemination has been actively studied [10, 5]. HCR is novel, in that it presents a simple dissemination framework that leverages node heterogeneity. We present a detailed summary of related work in Section 6 and conduct simulation studies to compare them against the various components of HCR wherever appropriate in Section 7.

2. HCR DESCRIPTION

We assume three classes of devices: sensors, accumulators and data sinks. Sensors are limited in both power and computational capacity, and are responsible for gathering information and transmitting to the nearest accumulator. We assume that sensors within a geographical distance of W, called the *communicable distance*, of each other can communicate. The choice of W depends on the deployment environment and is smaller than the actual wireless transmission range to account for shadowing, interference etc. We also assume that sensors have accurate estimates of their location.

An accumulator is equipped with more computational and transmission power and may also have privileged access to the shared wireless channel. An accumulator stores the output of the sensors belonging to its cluster for a limited time allowing it to compact the data received using application-specific algorithms, time-stamp the digest and transmit it to one or more data sinks. Data sinks are special accumulators that may be equipped with dedicated links, such as encrypted satellite up-links, to transmit sensor data globally.

The sensor network is laid out as a two level hierarchical cluster structure, $L0$ and $L1$[1]. An L0 cluster consists of sensors that are geographically located within hexagons of side $\frac{W}{\sqrt{13}}$. This choice for the hexagon size ensures that a node within an $L0$ cluster can communicate with nodes in its own and the six neighboring clusters. Periodically, nodes in an $L0$ cluster execute a power-aware, distributed algorithm to elect a cluster-head. The $L0$ cluster head is responsible for relaying all traffic that originates, passes through or terminates in the corresponding $L0$ cluster. Other sensors can "turn-off" their radios whenever they are idle.

$L1$ clusters are dynamically formed, "centered" at each accumulator and comprising all the $L0$ clusters for whom the accumulator is closest in terms of hop count. The accumulator at the center, acts as the cluster-head of the $L1$ cluster. For example, as shown in Fig. 1(a), the white and shaded areas are, respectively, two level-1 clusters with accumulators A_1 and A_2 as their level-1 cluster heads. The hop counts of each hexagonal grid are as shown in the figure. Each $L1$ cluster-head periodically broadcasts join messages asserting its presence and soliciting new $L0$ clusters to join the $L1$ cluster. We describe a novel constrained-forwarding technique that form $L1$ clusters using the minimum-possible number of messages. Further, the constrained-forwarding technique automatically constructs and updates routes from the sensors to the sink.

Once organized into hierarchical clusters, sensors send their gathered information to the respective $L0$ cluster heads which then relay data to the $L1$ cluster head (accumulator). The accumulator generates a single concise digest from these correlated, redundant data samples and sends the digest to a data sink. Routes from an accumulator to a sink may pass through multiple $L1$ clusters (Fig. 1(b)). Multiple routes of the same hop count may exist between any two accumulators. As discussed in Section 4, this multiplicity of paths, facilitates maintenance of network connectivity and improves robustness. Sensor failures are seamlessly handled by re-routing through alternate routes without need for an explicit route discovery phase.

3. CORE BUILDING BLOCKS

[1]Read level zero and level one

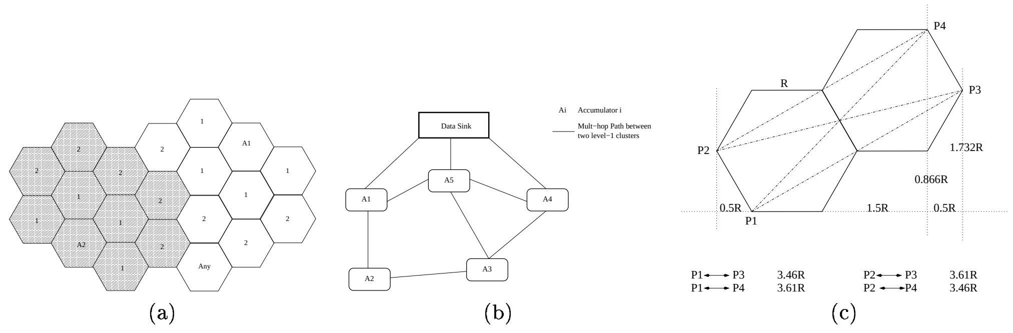
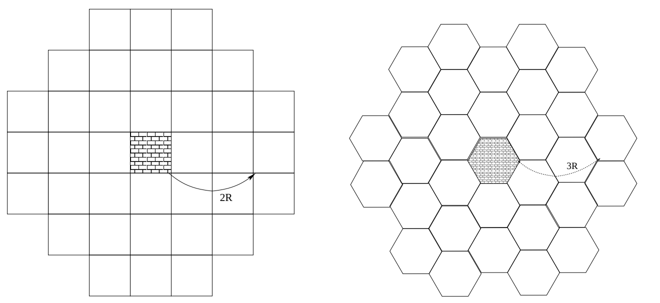


Figure 1: A two-level hierarchical clustering structure: (a) Two neighboring level-1 clusters and their associated level-0 clusters. The label of a hexagonal grid is its distance to the closest accumulator. (b) A network graph that is composed of level-1 clusters. Each "link" between two level-1 clusters represents a multi-hop path that traverses several level-0 clusters. (c) Determination of the size of the level-0 hexagonal grid.

In this section, we describe the core building blocks of HCR, namely the distributed election algorithm that forms $L0$ clusters, the constrained-forwarding technique that forms $L1$ clusters and routing within the HCR infrastructure.

3.1 Formation of $L0$ Clusters

Similar to GAF [8] in which square grids are formed, HCR imposes a two dimensional hexagonal grid structure. All the sensors in a hexagonal grid make up a level 0 cluster. As shown in Fig. 1(c), the maximum distance between any pair of devices in adjacent hexagonal grids of side R is $\sqrt{13} \times W$. Hence, we choose $R = \frac{W}{\sqrt{13}}$, such that devices in adjacent grids are within "communicable distance" of each other.



(a) Square grid: $R_{sq} = \frac{W}{\sqrt{5}}$, 37 interfering grids

(b) Hexagonal grid: $R_{hex} = \frac{W}{\sqrt{13}}$, 31 interfering grids

Figure 2: The number of grids in the vicinity that may conflict with a given grid. The grid in consideration is filled and all other grids in the vicinity that could conflict with this grid are shown. Improvement is $(\frac{37-31}{37}) = 16.22\%$ in terms of grids and $(1 - \frac{31 * A_{sq}}{37 * A_{hex}}) = 16.28\%$ in terms of geographical area where A_{sq} and A_{hex} are the areas of a square and hexagon respectively.

Recall that, if square grids are used the grid size needs to be set to $\frac{W}{\sqrt{5}}$ in order to maintain network connectivity[8]. Hexagonal level-0 clusters are an improvement for three reasons. First, intrinsic homogeneity of the hexagon allows communication with all the six neighboring grids, while a square grid can only communicate with grids sharing a common side[8]. Second, a network that is originally connected in the flat topology would become disconnected after clustering, only when all neighbors within communicable distance of a node, *no longer* lie within its own or neighboring grids. Hexagonal grids have a 25% smaller probability of network partition after clustering than square grids. Third, as shown in Figure 2, a hexagonal grid interferes with 16.22% fewer grids and over a 16.28% smaller area.

The leader election algorithm is initiated when any node fails to find the leader. This node sends out a leader proposal message announcing the duration of time that the sender agrees to be the leader T_a.

$$\text{Accumulator: } T_a = \infty, \quad \text{Sensor: } T_a = y \times Power_{rem}. \quad (1)$$

If a leader already exists, and intends to be the leader for longer than T_a, it immediately sends out a leader proposal of its own. If no leader exists, each sensor device responds to the proposal, only if it intends to stay the leader for longer than T_a. Since there could be more than one such node, responses are delayed such that nodes with larger remaining power announce themselves earlier. The delay is as follows:

$$Delay = \frac{x}{Power_{rem}} + R + S, \quad (2)$$

where $0 \leq R, S_{sensor} \leq \frac{x}{2 * MaxPower}], S_{accum} = 0$. Note that the randomized part of the delay is necessary to avoid collisions and is an order of magnitude smaller than the deterministic part. This ensures that the delay is always dominated by the deterministic part. Leader election terminates when the node with the largest value of T_a makes its proposal, at which time all the other nodes "turn-off" their radios for a maximum duration of T_a. When one or more of these nodes wake-up, the leader-election starts anew.

The leader election algorithm has several interesting properties, (i) If an accumulator is present in the $L0$ cluster, it would always win the election, on account of smaller delay to respond and larger T_a. (ii) Delaying the response to a leader proposal ensures that leader election uses only two messages as the best proposal is transmitted the earliest. (iii) If no accumulator exists, each sensor serves as the leader for a small amount of time, after which another node takes over, on account of higher remaining power at the other nodes. Thus, the load of relaying cross-traffic is balanced among the various sensors in a $L0$ cluster. Finally (iv) each sensor in a $L0$ cluster can use a lower transmission range of $2R$, instead of $\sqrt{13}R$ as it only needs to communicate with neighboring grids.

3.2 Formation of Level-1 Clusters

While, $L0$ clusters are statically formed using geographical information, $L1$ clusters, are dynamically formed in a decentralized manner to exploit node heterogeneity. $L1$ clusters are centered at resource-rich accumulators to direct sensor data transmission toward the closest data processing entity. The cluster formation mechanism described below also establishes routing state.

3.2.1 *Forwarding of* join *Messages*

Once an accumulator A_1 is elected as the $L0$ cluster head of its local hexagon, it broadcasts a join message to its adjacent $L0$ clusters. The join message contains, A_1's identifier and the hop count from A_1, initialized to 1. Upon receipt of a join message, an $L0$ cluster head checks if it has received a join message from any accumulator with a smaller hop count. If not, it updates its (accumulator id, hop count) record and forwards the join message to its neighbors.

To reduce the number of join messages exchanged, we note that:

(1) A join message should *never* be forwarded to nodes that already know a better path, such as the originator or the left/right neighbors of the originator. To achieve this, the join message is extended to contain the grid identifier of the originating $L0$ cluster. Upon receipt of a join message, an $L0$ cluster, identifies the originator and forwards the message to the three clusters in the opposite direction as shown in Fig. 3 (a).

(2) A join message advertising a longer path should *never* be forwarded[9]. If a join message advertising a longer path arrives earlier at an $L0$ cluster than the one advertising the best path, the $L0$ cluster head may forward the first message to all subsequent $L0$ clusters, which in turn, forward it outward. Although, this does not impair the correctness of the algorithm, significant power is consumed in sending unnecessary join messages.

To alleviate this problem, the join message is extended to include the hop counts of the forwarding $L0$ cluster's two neighbors that are also adjacent to the cluster to which the message is being forwarded. In summary, a join message has the following fields: (i) **A** : the Accumulator Identifier; (ii) **H** : the advertised hop count; (iii) **G** : the identifier of the forwarding $L0$ cluster (grid); (iv) $\mathbf{H_r}, \mathbf{H_\ell}$: the hop counts of the common neighbors of the sending and receiving grids, where the right and left neighbors are defined with respect to the sender and (v) **S**: the sequence number. With the above information, an $L0$ cluster head, R, determines whether or not to forward a join message to its neighbors, using the following rules.

$$R \rightarrow X : \textbf{if } H_\ell = H \textbf{ then } H_{advt} = H + 1, \textsf{forward } \textbf{else} \textsf{ drop} \quad (3)$$

$$R \rightarrow Y : H_{advt} = H + 1, \textsf{forward} \quad (4)$$

$$R \rightarrow Z : \textbf{if } H_r = H \textbf{ then } H_{advt} = H + 1, \textsf{forward } \textbf{else} \textsf{ drop} \quad (5)$$

Figure 3(a) depicts how join messages are being forwarded. When the cluster head marked 'Y' receives message 'a', with $H_{advt} = 1$, it knows that the northern neighbor 'B' also has a hop-count of '1' and so forwards a join message onward to the neighboring grid on the north-eastern side 'X'. Now when this grid 'X' receives the message 'b', with $H_{advt} = 2$, it realizes that the neighbor to the north-west 'B' has a lower hop count than itself and so desists from forwarding a message to its northern neighbor 'Z'. We prove formally in [9] that such constrained-forwarding ensures join messages advertising longer paths are never forwarded.

The state kept at each $L0$ cluster head is soft, and each $L1$ cluster head periodically re-transmits a join message so that message losses will not adversely affect the correctness of the protocol. Each $L0$ cluster maintains "liveness" information regarding its neighboring hexagons either by snooping at the MAC layer or by a periodic exchange of $HELO$ packets. A hexagonal grid is said to be live if at least one sensor exists in that grid. Liveness information is used to route around holes as follows: (i) join messages are forwarded only to "live" neighbors; (ii) the hop count of a "dead" grid is infinite when join messages are exchanged between grids that are both neighbors of the "dead" grid. This allows the constrained-forwarding technique to by-pass holes at the lower layer as shown in Figure 3(b).

As a further optimization, instead of dispatching individual join messages to each of its neighboring grids, a cluster head may perform all the forwarding operations by a single broadcast. Specifically, the broadcast message contains bit-masks that advertise whether or not a specific neighbor needs to act upon the receipt of this join message (6 bits, one for each neighbor). Further, as proved in Section 5 the values of H_r, H_ℓ are either $H, H-1$ or $H+1$, when H is the hop count of the current grid. Thus, instead of sending out the actual values for each neighbor, a cluster head sends only the value H and 4 bits per neighbor, wherein 2 bits encode the "nature" of each hop-count value as there are three possible variants from the base value of H. This reduces the number of messages used and also the size of each message. Each join message is 8 bytes long.

3.2.2 *Creation of Level-1 Clusters*

The process of forwarding join messages originated by accumulator A_1 terminates when the message reaches an $L0$ cluster with a smaller hop count to a different accumulator A_2 (Fig. 3(c)). The recipient of the join message discards the message, and returns a **N**eighbor **A**c**K**nolwedgement (NAK) message toward the sender cluster. The NAK message contains the same information as a hypothetical join message of the other accumulator. Upon receipt of a NAK message, an $L0$ cluster head realizes that it is a border grid of the originating accumulator A_1, and forwards a **B**order **R**outer **N**otification (BRN) message back to A_1 via its upstream neighbor grids. The BRN message is of 5 bytes and has the following fields, (i) $\mathbf{A_2}$: the identifier of the adjacent accumulator ($L1$ cluster head); (ii) **B**: the identifier of the level-0 border grid; (iii) $\mathbf{H_1}$: the hop count from B to the original accumulator A_1; (iv) $\mathbf{H_2}$: the hop count from B to the adjacent accumulator A_2; and (v) **S**: the sequence number.

All the level-0 clusters that forward the BRN message to the accumulator maintain a list of accessible border grids and, for each border grid, the list of neighboring grids that can be used to reach this border grid. As will be discussed in Section 4, this facilitates routing between accumulators. Upon receipt of a BRN message, an accumulator keeps track of the list of its adjacent $L1$ clusters and for each adjacent $L1$ cluster, the set of associated border grids. Since, state maintained

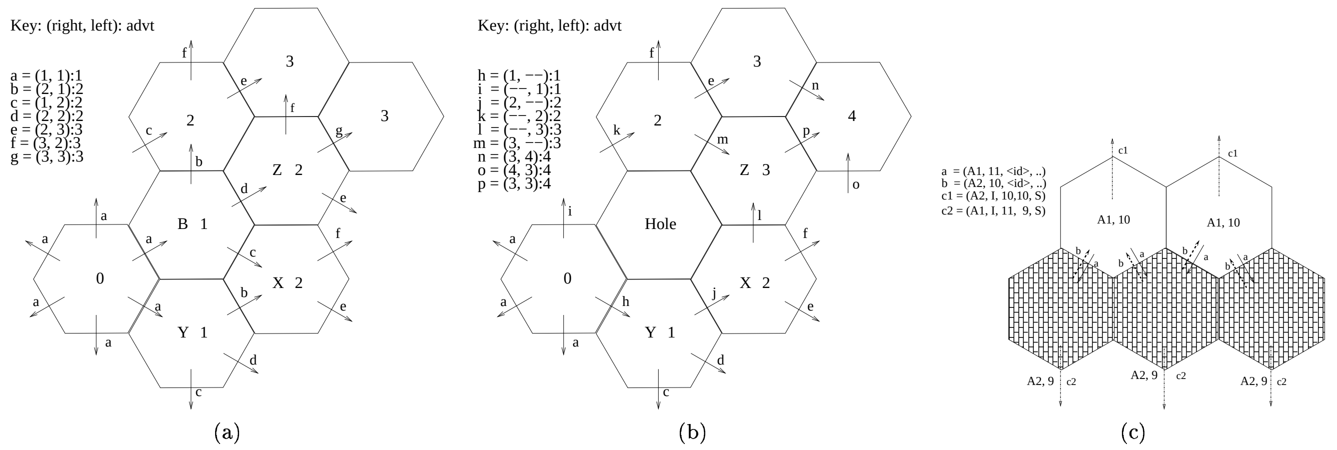


Figure 3: (a) An illustration that demonstrates how join messages are being forwarded. (b) Managing holes: Both neighbors forward "extra" messages to Z. (c) Termination of the join message forwarding process

by $L0$ cluster heads is soft and join messages are transmitted periodically, the $L0$ cluster heads rejoin a different $L1$ cluster should their current accumulator fail. Further, as detailed in [9] the accumulators need not synchronize the transmission of join messages.

3.3 Properties of the Proposed Cluster Formation Algorithm

P1 Each $L0$ cluster head, except the originating accumulator, forwards a join message to at least two distinct grids and at most three distinct grids; and each $L0$ cluster head receives a join message from at least one grid and at most two distinct grids.

P2 The number of join messages that are sent during the formation of a $L1$ cluster is bounded and is of $O(n)$, where n is the number of grids in that cluster. In comparison, an unconstrained forwarding algorithm may result in $O(n^2)$ messages.

P3 In the case that an accumulator initiates its cluster formation procedure much earlier than another accumulator, the former may include more grids in its cluster than it ought to, and the later accumulator would have to reclaim those grids. However, as a grid can at most have six accumulators contend for it, The total number of join messages exchanged during the *entire* cluster formation phase is bounded by $O(N)$, where N is the total number of live hexagonal grids. This is the best possible for any cluster formation mechanism.

4. ROUTING IN THE TWO-LEVEL CLUSTER ARCHITECTURE

HCR integrates cluster formation with route update and maintenance, thereby eliminating routing overhead. Due to space constraints, routing in the two-level cluster topology is briefly described. A complete description is available[9].

Within an $L0$ cluster

Sensors within an $L0$ cluster forward generated data, in a single hop, to their $L0$ cluster head, which forwards raw data toward the $L1$ cluster head.

Within an $L1$ cluster

Routes necessary for transmission of both upstream and downstream traffic are maintained and updated through join, BRN, and NAK messages. Specifically,

(i) Upon receipt of a valid join message that advertises a better hop-count, an $L0$ cluster head stores an upstream route to its $L1$ cluster head, through the associated upstream neighbor $L0$ cluster head(s) that forwarded this message.

(ii) Upon receipt of a valid BRN message from a downstream neighbor, an $L0$ cluster head records route to a neighbor $L1$ clusters through the associated downstream neighbor. An accumulator uses the BRN message to record neighboring $L1$ clusters.

(iii) Upon receipt of a NAK message, an $L0$ cluster head identifies itself as a border cluster and keeps track of a list of adjacent $L1$ clusters and the neighbor grid to reach each adjacent $L1$ cluster.

Across level-1 clusters

Each accumulator keeps track of (i) a list of adjacent $L1$ clusters; (ii) for each adjacent level-1 cluster, a list of $L0$ border clusters (grids) to reach the adjacent accumulator, the cost to reach the adjacent $L1$ cluster, and the route to sink from the adjacent $L1$ cluster; and (iii) for each border cluster, the list of immediate, downstream neighbor $L0$ grids that can be used to reach that border cluster.

While most of this information can be collected and updated from the BRN messages received (Section 3), accumulators communicate with one another to identify routes to the sink. To send a message to an adjacent accumulator or a data sink, an accumulator chooses a $L0$ border cluster and forwards the message to the corresponding $L0$ neighbor(item (ii)). Interior nodes use downstream routes to forward this data. Upon receipt of a message, an $L0$ border cluster forwards it to one of its neighboring $L0$ clusters that belong to the destination $L1$ cluster. On receiving a cross-border message, an $L0$ border cluster forwards it to its accumulator as if it were sensor information generated by this grid.

The proposed hierarchical cluster structure handles the existence of “empty” grids using alternate paths that by-pass the empty grids that are found during message forwarding (Figure 3(b)).

5. PROPERTIES OF TWO-LEVEL CLUSTERING AND THEIR FORMAL PROOFS

We prove several properties stated in Sections 3.1 and 3.3. We first prove in Theorem 1 that the probability of network partition as a result of clustering is smaller under hexagonal grids than under square grids.

THEOREM 1. *Let p_r and p_h denote, respectively, the probability that a pair of sensors within the wireless transmission range of each other remains in neighboring clusters under square grids and hexagonal grids. Then, assuming uniform density, $p_h = 1.4 \times p_r$.*

Proof: The probability that a pair of sensors within the wireless transmission range of each other remains in neighboring clusters is

$$p = \frac{\text{Communicable area in a grid scheme}}{\text{Communicable area in a flat network}}.$$

Two facts are sufficient to conclude. First, notice that a node in a square grid can communicate with nodes in four neighboring grids, while a node in a hexagonal grid can communicate with nodes in six neighboring grids. Second, square grids have side $\frac{W}{\sqrt{5}}$ while hexagonal grids have $\frac{W}{\sqrt{13}}$. □

We now establish the theoretical base of **P1** with Theorems 2 and Corollary 1. For notational convenience, we define the *label* of a $L0$ grid to be the shortest distance (in terms of hop count) from itself to the closest accumulator.

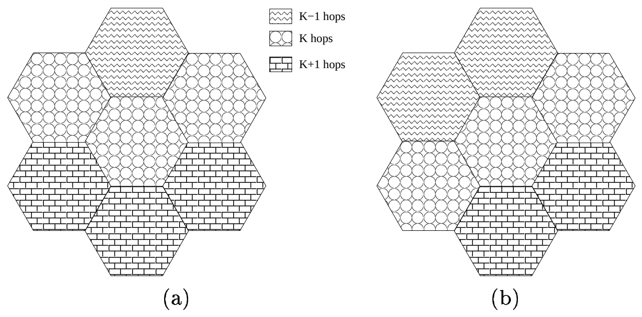


Figure 4: The type of grids that can exist. (a) Type 1, (b) Type 2 and mirror of Type 3

THEOREM 2. *Under the assumption that there exists at least one sensor in each hexagonal grid, each non-accumulator grid is one of the three types shown in Figure 4.*

Proof: The geometric intuition is that the geometric centers of grids that have a label k for any $k \geq 1$ form a much larger hexagon centered at the accumulator. Grids at the corners of this larger hexagon are of type 1, while other grids are of type 2 or its mirror image. An algebraic proof is in [9]. □

THEOREM 3. *Every* join *message, advertising hop-count $n+1$, originates in a grid with a label of n and is forwarded to a grid with a label of $n+1$.*

Proof: We prove this by induction on the hop-count advertised in a join message. The forwarding rules specified in Equations 3-5 are used in the inductive step. A detailed proof is in [9]. □

Noticing that grids of Type 1(2,3) have three(two) neighboring grids with the appropriate labels leads to the following corollaries.

COROLLARY 1. *Each level-0 cluster head receives a* join *message from at least one grid and at most two distinct grids.*

COROLLARY 2. *Each level-0 cluster head, except the originating accumulator, forwards a* join *message to at least two distinct grids and at most three distinct grids.*

Property **P1** summarizes Corollaries 1 and 2. Properties **P2** and **P3** follow from Theorem 4 which is proved in [9].

THEOREM 4. *The number of* join *messages that are sent within a $L1$ cluster during cluster formation is bounded and is of $O(n)$, where n is the number of grids in a level-1 cluster. Moreover, the total number of* join *messages exchanged is bounded by $O(N)$, where N is the total number of live hexagonal grids.*

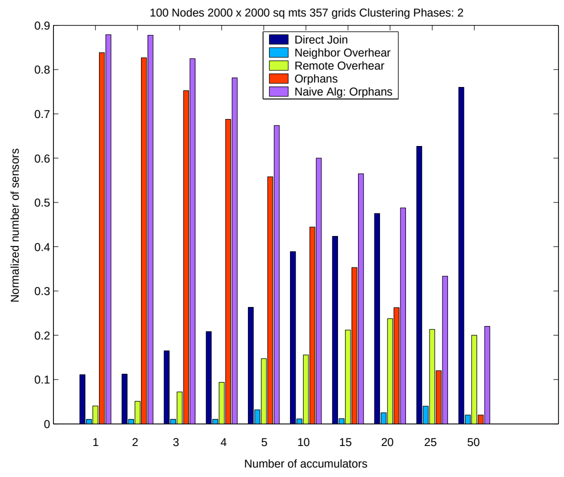


Figure 5: Normalized number of nodes that succeeded to form clusters after two runs of HCR vs. different accumulator-to-sensor ratios. Also shown are the number of orphan nodes under HCR and the naive algorithm.

6. RELATED WORK

Node heterogeneity imposes certain added constraints to cluster formation, such as (i) need for multi-hop clusters due to the ad-hoc nature of accumulator placement and (ii) clusters need to be centered/rooted at accumulators. To the best of our knowledge, none of the existing clustering algorithms embrace the notion of node heterogeneity. However, several individual components of HCR are related to existing work. A more detailed discussion is in [9].

Clustering and routing

In k-cluster-based routing [7], the network is dynamically clustered, such that nodes in a cluster can be reached from any other node within the cluster in k hops and routes are computed using the Dijkstra's shortest path algorithm. HCR's $L1$ clusters, on the other hand, are constrained in the choice of cluster centers but not in the diameter of a cluster. ZRP [4] enables nodes to maintain *routing zones* consisting of nodes that are at most n hops away and route packets using a hybrid strategy of proactive intra-zone routing and demand-based inter-zone routing. The spine routing framework [3] was built upon the notion of *spines* (virtual backbones) such that every node is either part of the spine or one hop away from a node in the spine. Although this work presents a dedicated backbone for information dissemination, spine maintenance and routing is costly and introduces significant control traffic when updates are made. The HCR architecture exploits the geometric property of level-0 clusters to devise a simpler, much lighter weight routing mechanism.

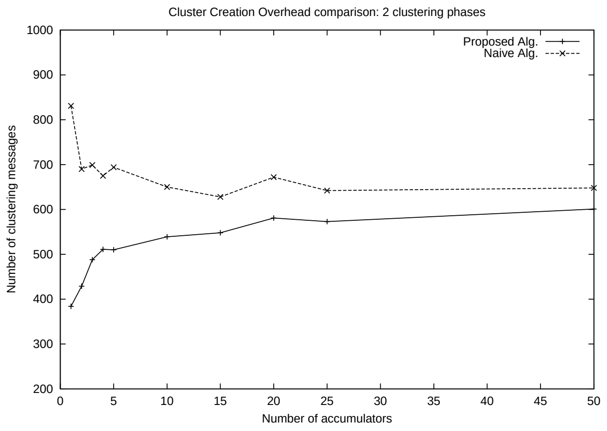


Figure 6: Message overhead incurred in cluster formation. HCR's constrained forwarding technique reduces message overhead by 120-210%. 500 Nodes are distributed over $4km^2$.

Power Saving Techniques

Span [2] is a distributed and randomized algorithm, that does not require the support of GPS. Nodes exchange HELLO messages to determine whether they should sleep or join a forwarding backbone. Span-based clusters essentially play the same role as $L0$ clusters in HCR, but take much longer to form and are complex to organize into an $L1$ cluster rooted at the accumulator. GAF[8], which motivates our design of $L0$ clusters, assumes the availability of GPS and conserves energy by dividing a region into square grids. HCR uses hexagons instead, as hexagonal grids cause much lower interference and have a greater probability of retaining connectivity.

Dissemination Architectures

TTDD [10] and Diffusion [5] are two interesting dissemination architectures for homogeneous sensor networks. While Diffusion builds sink-based dissemination trees, which the sources join to send data towards the sink, TTDD builds greedy dissemination trees for each source and specifies mechanisms by which a sink can join and leave the dissemination trees of the source(s) it is interested in. HCR builds a dissemination infrastructure that is a hybrid between these two ideologies. At level-1, $L1$ clusters are essentially data-collection trees rooted at an accumulator which allow the resource-rich accumulators to act as effective data sources to the sinks. Accumulators can either build TTDD-like greedy dissemination trees or join Diffusion-like dissemination trees built by the eventual data sink. In this sense, HCR suggests an alternative but complementary dissemination architecture.

7. SIMULATION RESULTS

We have performed an ns-2 simulation study in order to quantify the performance improvements provided by resource heterogeneity which are briefly summarized here. A much broader study is present [9].

7.1 Methodology

A pre-defined hexagonal grid map of side $\frac{W}{\sqrt{13}}$ was superimposed on a rectangular flat terrain area with size varying between $.36\ km^2$ to $4\ km^2$. The wireless range of each sensor was assumed to be 250m. N nodes were distributed at random over the rectangular region, out of which N_a nodes were randomly selected as accumulators, where N ranges from 100 to 500 and N_a from 1 to 50. For all simulations wherein appropriate, accumulators are assumed to perform a 50% data compression, i.e., 1 digest packet is generated for every 2 raw packets. One of the N nodes was randomly chosen to be the sink, while data sources were randomly chosen from non-accumulator nodes. Each data source was configured with a constant bit rate traffic generator working above an UDP agent. Protocol agents that participate in cluster formation and route establishment/maintenance were deployed at each of these nodes. The cluster formation process repeats periodically with a period of 30 seconds.

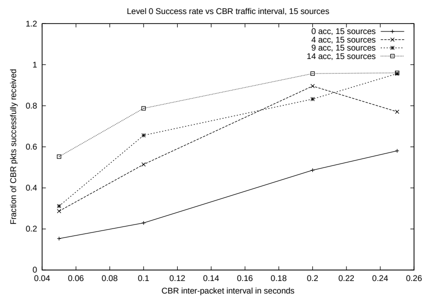


Figure 7: Improvement of data throughput with cluster formation. Data throughput, counted as the fraction of total messages delivered, improves by upto 200%

7.2 Performance of HCR

Successful rate of cluster formation:

HCR reduces control message overhead by eliminating redundant and incorrect join messages. A 2% density of accumulators is enough for HCR's cluster formation to successfully cover all the sensor nodes in all but the sparsest topology [9]. In a very sparse distribution of 100 nodes in a $4km^2$ area, Figure 5 shows that HCR has a much higher success rate than the naive forwarding algorithm. We also introduce two additional heuristics (a) Neighbor overhear and (b) Remote overhear that allows nodes to conservatively snoop join messages in their vicinity [9] that further increase the successful rate of clustering.

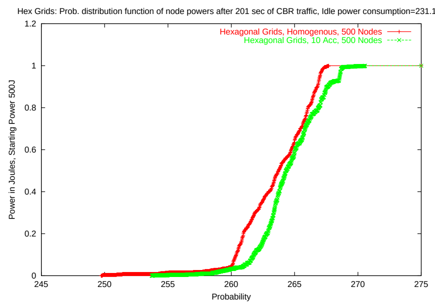
Message overhead of $L1$ cluster formation:

Figure 6 depicts the number of control messages incurred during cluster formation under HCR and the naive algorithm. Recall that the naive algorithm could create as many as $O(n^2)$ messages, as opposed to the $O(n)$ bound by HCR. As shown in Figure 6(a), the message overhead incurred under the naive algorithm is between 120% to 210% of that under HCR.

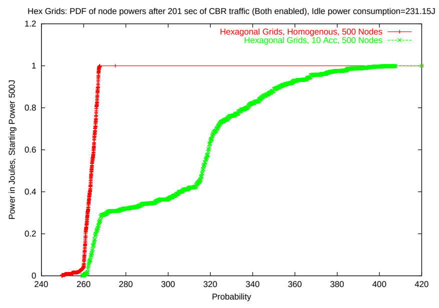
7.3 Improvements Due to Heterogeneity

Increase in Data Throughput

Figure 7 shows that HCR improves sensor-sink data throughput by upto 200%. This is because HCR uses heterogeneous accumulators to process sensor data close to the point of origin, thereby decreasing network resources spent on redundant information. We experimented with a wide range of data rates and source density [9] but show only one.



(a)



(b)

Figure 8: Power savings due to heterogeneity. (a) Decrease in long-haul traffic lowers power spent in transmit/receive states, (b) $L0$ clusters allow non-leader nodes to turn-off radios and save power spent in idle state.

Power savings

We now consider power saving achieved by the deployment of heterogeneous nodes. 500 nodes are deployed randomly in a 4 km^2 region. A randomly chosen subset of 15 nodes act as CBR sources. We compare the probability distribution function of remaining power at each node for the homogeneous case against that for the case wherein 10 accumulators are present randomly in the topology. HCR's $L1$ clusters lower the power spent in transmitting/receiving data by upto 30% while the $L0$ clusters allow non-leader nodes to turn off their radios and accrue much larger savings. The unique shape of the pdf is due to the different rates of power drain on sensors that are either close to an accumulator or on the path between accumulators as opposed to those that are away from such "busy" routes [9].

7.4 Comparison between HCR and Other Clustering Schemes and Power Savings

We compare HCR with other clustering schemes on the efficiency of cluster formation and power savings accrued for specific traffic patterns [9]. Our result, omitted for lack of space, show that in similar heterogeneous deployments, HCR requires upto 10% fewer messages and provides upto 18% more power savings.

8. CONCLUSION AND FUTURE WORK

In this paper, we demonstrate algorithms that can effectively organize sensor networks consisting of a small density (<5%) of resource-rich nodes, in order to improve data throughputs by up to 200% and reduce energy consumption during packet transmission significantly (30% at very low data rates). We demonstrate that a broadcast-based cluster formation algorithm, that works on top of a static hexagonal grid layout, preserves network connectivity in networks as sparse as 100 nodes in 4 km^2 (wireless range 250m). Further, we introduce constrained forwarding optimizations that are shown analytically and through simulations to achieve significant reductions in message overheads and substantially greater coverage, i.e. more nodes successfully join the cluster.

We identify as future research, capacity analysis of traditional sensor networks that are usually multiple-source single-sink ad-hoc networks. While several relevant problems such as random source-destination pairs and single source-destination pairs have been already solved, this problem presents unique challenges and hasn't been discussed to the best of our knowledge.

9. REFERENCES

[1] S. Bandyopadhyay *et. al.* "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," in Infocom 2003.

[2] B. Chen *et. al.* "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in Mobicom, 2001.

[3] B. Das *et. al.* "The Clade Vertebrate: Spines and Routing in Ad-Hoc Networks," in ICC, 1998.

[4] Z. Haas, "A New Routing Protocol for The Reconfigurable Wireless Networks," *In Proc. of the IEEE International Conference on Universal Personal Communications,* October 1997.

[5] C. Intanagonwiwat *et. al.* "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks" in Mobicom, 2000.

[6] C. R. Lin *et. al.*, "Adaptive Clustering for Mobile Wireless Networks," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 7, September 1997.

[7] N. H. Vaidya *et. al.*, "A Cluster-Based Approach for Routing in Dynamic Networks," *ACM SIGCOMM Computer Communication Review,* pp. 49-65, April 1997.

[8] Y. Xu *et. al.* "Geography-informed Energy Conservation for Ad Hoc Routing, in Mobicom, 2001.

[9] S. Kandula *et. al.* "A Case for Resource Heterogeneity in Large Sensor Networks" UIUCDCS-R-2003-2233.

[10] F. Ye *et. al.* "A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks" in Mobicom, 2002.