Design and Analysis of Hybrid Indirect Transmissions (HIT) for Data Gathering in Wireless Micro Sensor Networks*

Benjamin J. Culpepper[a] **Lan Dung** **Melody Moh***

Department of Computer Science, San Jose State University, San Jose, CA, USA
[a] Neuro-Engineering Laboratory, NASA Ames Research Center, Moffet Field, CA, USA

Sensor networks have many potential applications in biology, physics, medicine, and the military. One major challenge in sensor networks is to maximize network life under the constraint of limited power supply. The paper addresses energy-efficiency in the context of routing and data gathering. A new protocol is proposed: Hybrid Indirect Transmission (HIT). HIT is based on a hybrid architecture that consists of one or more clusters, each of which is based on multiple, multi-hop indirect transmissions. In order to minimize both energy consumption and network delay, parallel transmissions are used both among multiple clusters and within a cluster. This is made possible by having each sensor independently compute a medium access controlling TDMA schedule. The computation within each sensor is intelligent yet simple. Formal analysis shows that it requires O(n) space and O(n× logn) time complexities, and O(1) setup messages prior to the computation, where n is the total number of sensors. HIT does not require sensor nodes with CDMA capability, or the remote base station to compute a data gathering schedule. Performance is evaluated by simulating and comparing HIT with three other existing protocols, including Low Energy Adaptive Clustering Hierarchy (LEACH), Power Efficient Gathering for Sensor Information System (PEGASIS), and Direct Transmission. Results have shown that HIT greatly reduces both energy consumption and network delay; it also maintains longer network life compared to these three existing protocols. Security issues and a potential application of HIT in biomedical sensing technology are also rigorously discussed. This work is significant to the advancement of energy-efficient micro sensor networks; the proposed protocol is promising and would contribute to the use of wireless micro sensor networks in future biomedical sensing technologies.

I. Introduction

Miniaturization through advances in micro electrical-mechanical (MEMS) technology have led to the development of micro sensors that have their own power, processing, and communications systems in a compact package. Sensor networks can be used in the military to monitor troop movements [24], in biology to study animal behavior [14], and in civil engineering to monitor the health of large structures, such as bridges and freeway overpasses [11].

The goal of many micro-sensor projects, including Smart Dust [19], is to build cubic millimeter sensors. Each micro sensor is self-sufficient. They have their own operating systems, they can make their own decisions, and they can generate power. Micro sensor networks leverage their numbers to create a robust, fault-tolerant, low-cost, and accurate sensing network.

The small size of micro sensors limits the amount of energy available for tasks such as data processing and communications. State-of-the-art battery technologies store one joule in a square millimeter, and the latest solar cells are capable of generating one joule per day per square millimeter of surface area [19].

The major consumer of power in sensor networks is communication [8]. In the radio model commonly used for sensor networks, the most expensive operation is usually transmission, since the energy cost of

* A preliminary version of the paper was presented at the IEEE Computer Communication Workshop, October 2003. Melody Moh is the correspondence author and is supported in part by CSUPERB (California State Univ. Program for Education and Research in Biotechnology). *Email: moh@cs.sjsu.edu*

a transmission increases with the square of the distance the message is broadcast [8,13]. Because of this, one of the more energy-expensive applications for a sensor network is *remote sensing*, in which a user monitors a distant environment. The sensor network's task is to sense the environment, and relay the information back to a *remote* base station where the user can access it. Reducing the energy consumption and number of transmissions are therefore important design criteria in these sensor networks.

It might seem natural to consider using ad-hoc routing protocols for sensor network applications. However, they were not designed with the requirements of sensor networks in mind. Like some ad-hoc networks, sensor network protocols must be power aware [12]. Ad-hoc routing protocols, however, do not take into consideration sensor energy levels, the costs of long-term routing table storage, in-network data processing, or cooperative dissemination [18]. It is usually necessary to perform significant adaptation to an ad-hoc protocol before it can run efficiently on a sensor network [20].

The purpose of this paper is to introduce *Hybrid Indirect Transmission (HIT)*, a novel routing protocol for sensor networks that increases network lifetime. The major features of HIT include (1) utilizing one or more clusters to reduce the number of transmissions to the remote base station, and (2) parallel, multi-hop indirect transmissions even in the presence of multiple, adjacent clusters. These features greatly reduce energy consumption and network delay.

HIT has been designed for use in bioelectric computer interfaces, specifically for sensing electromyogram (EMG) and electro-encephalogram (EEG) signals [18]. In terms of data model, network size, data fusion, and energy level, HIT matches the major requirements and conditions of many potential applications in biomedical sensing.

The paper is organized as follows. Section II presents background and related studies. Section III discusses the basic models that HIT is based on. Section IV presents the details of HIT, including discussions on fault tolerance mechanisms, and a formal analysis. Detailed proofs are in the Appendix. An extensive set of performance analysis simulation experiments is presented in Section V. Section VI addresses the security issues of HIT. Section VII presents the current deployment of sensor networks, and details a potential application of HIT in the area of biomedical sensing technology. Finally, Section VIII concludes the paper and outlines directions for future work.

II. Background and Related Studies

Current remote sensing routing protocols increase efficiency through data fusion, power management systems, clustering, and chaining. Data fusion reduces packet size, while clustering and chaining minimize transmission costs.

A clustering protocol segments a network into non-over lapping clusters, each of which is led by a cluster-head. Data sensed by non-cluster-heads is sent to the cluster-head, where it is fused, then transmitted to the base station. Clustering protocols distinguish themselves by how they elect cluster-heads. The distributed clustering algorithm (DCA) [2] and the weighted clustering algorithm (WCA) [3] for mobile ad-hoc networks use weights to select cluster-heads. The worst-case time complexity of WCA is $O(N^2)$. However, this work applies mainly to mobile ad-hoc networks, and does not consider the specific limitations of micro sensor networks.

Many protocols use heuristics to elect cluster-heads [6,22]. These heuristics are based on the minimization of both transmission distances and the number of cluster-heads [22]. Ghiasi *et al* developed an optimal clustering algorithm to select cluster-heads using load balancing, so that each cluster has the same number of nodes and distances between nodes and cluster-heads are minimized [6]. The algorithm requires position knowledge of all sensors, and takes $O(N^3)$ time.

The LEACH protocol requires no position knowledge and uses self-election, where each node has a probability p of becoming a cluster-head. It guarantees that every node will be cluster-head only once in $1/p$ rounds [8]. Bandyopadhyay *et al* developed a formula for predicting an optimal p based on a simplified model of a LEACH network, where the base station was placed at the center of the network [1].

Another class of remote sensing routing protocols is the chaining protocols. In a chaining protocol, nodes form a linear network and only transmit to close neighbors. Included in this class are Power Efficient Gathering in Sensor Information Systems (PEGASIS) and chain hierarchy protocol [13]. In the PEGASIS protocol, a chain of nodes is computed using a greedy algorithm. During data gathering, chain leaders are elected to fuse data from the network and transmit the result to the base station. Chain setup takes $O(N)$ in both time and space.

The performance differences between chaining and clustering can best be understood by comparing the LEACH and PEGASIS protocols. The two advantages of using clustering in LEACH are in-

creased energy efficiency, since only the cluster leaders communicate to the base station, and parallelism, since each cluster is assigned a CDMA code, allowing simultaneous communications among adjacent clusters. The PEGASIS protocol uses chaining to increase energy efficiency over LEACH at the expense of increased delay; long delays occur in PEGASIS, because sensors must transmit one at a time, whereas in LEACH the number of simultaneous transmissions is equal to the number of cluster-heads. *HIT combines the advantages of both protocols by allowing simultaneous transmissions both among clusters and within one cluster, without requiring prior position knowledge. Additionally, HIT chains, and therefore network delays, are shorter.*

Younis *et al* compared routing protocols within a cluster, and found that the LEACH-like routing protocol called *minimum distance* was the poorest performing protocol in terms of energy efficiency [22]. Their network infrastructure was slightly different than in LEACH, with the addition of specialized nodes called gateway nodes, which act as cluster-heads. The best performing protocol in terms of energy efficiency was minimum distance squared, which uses multiple PEGASIS-like chains inside a cluster to reduce energy costs. This finding supports the superiority of the HIT protocol presented in this paper. Their protocol, however, requires gateway nodes with position knowledge to compute routes and arbitrate medium access among sensors, whereas *in HIT the routing and medium access (scheduling) are computed independently by each sensor without prior position knowledge.*

III. Basic Models

This section presents the basic models underlying HIT. The data delivery model is presented, and proof of how HIT routes always save energy over direct transmissions is given. Finally, the importance of using parallel transmissions is addressed, and a brief argument for TDMA's superiority over CSMA/CD for continuous data gathering is made.

III.A. Data Delivery Model

Sensor networks can be classified in terms of data delivery required by the application interest into the following: 1) continuous, 2) event-driven, 3) observer-initiated, and 4) hybrid [18]. These models govern the generation of the application traffic. The proposed HIT is based on the continuous model, which assumes that sensors always have data to transmit, and the data are continuously sent at a prespecified rate.

In addition to data delivery from the application perspective, we must also consider the actual flow of data packets between the sensors and the observer; this is a routing problem subject to the network protocol [18]. HIT uses data aggregation techniques that are plausible in several widely studied applications [7,8], and reduce the overhead of a broadcast approach for the flow of data from the sensors to the observer. The major advantage of broadcast is that it does not rely on a complex network layer protocol for routing, addressing, and location management; existing sensor network efforts have mostly relied on this approach [7].

III.B. Radio Model

The radio model used in this paper is the first order radio model [8,13]. In this model a radio transceiver dissipates E_{elec} = 50 nJ/bit and uses $\in_{amp}$ = 100pJ/bit/m^2 to run the transmitter amplifier. The transmitter is assumed to have variable power control. The channel is assumed to be symmetric, so the power required to transmit a message from A to B is the same as that required to transmit from B to A. Table 1 shows the model's energy dissipation of reception and transmission for a *k*-bit message over distance *d*.

Operation	Energy Dissipation
Transmission	$E_{tx}(k,d) = E_{elec} * k + \in_{amp} * k * d^2$
Reception	$E_{rx}(k) = E_{elec} * k$

Table 1. First order radio model.

III.C. Analysis of Direct and Indirect Transmissions

In a cluster-based protocol such as LEACH, each member transmits information *directly* to the cluster head. By contrast, HIT nodes transmit *indirectly* to the cluster head by using a multi-hop route from each node to the cluster-head (see Figure 1).

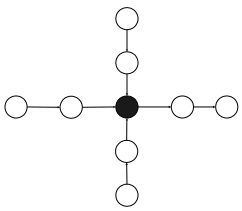

Figure 1. A HIT cluster consisting of a cluster head and 4 chains.

Previous works have shown that indirect transmissions do not always result in energy savings [8,16]. Consider Figure 2 where node 1 has a *k*-bit packet to send to the cluster-head at distance *nr*. Assuming our first order radio model, if node 1

chooses to send the packet directly, then the energy cost to the network will equal the sum of transmission by node 1, and reception by the cluster-head:

$$(E_{elec} * k + \in_{amp} * k * (nr)^2) + (E_{elec} * k)$$

If node 1 chooses to send the packet indirectly, through each intermediary node, the energy cost to the network will equal the sum of *n* transmissions over distance *r*, and *n* receptions:

$$n * (E_{elec} * k + \in_{amp} * k * r^2) + n * (E_{elec} * k)$$

It is easy to see that because of the additional energy involved in reception, as *r* decreases, there is a point at which it is more energy-efficient for node 1 to transmit directly.

However, the above analysis does not apply directly to HIT, mainly because it does not consider data fusion. HIT assumes that each intermediary node has its own packet that it wishes to transmit to the cluster-head, and these packets can be fused together, resulting in a packet that is only *k*-bits in size (or some small linear multiple thereof). Although there is an energy cost associated with data fusion, it is negligible compared to that of transmission and reception [8].

Let us now compare the energy costs incurred to the network by both direct and indirect transmission schemes, when each node has a packet to send to the cluster-head, and data items can be fused in the aforementioned manner. Direct transmission costs the sum of *n* transmissions as the distance increases from *r* to *nr*, plus *n* receptions:

$$\sum_{d=1}^{n} \left[\mathrm{E}_{\mathrm{elec}} * \mathrm{k} + \in_{amp} * k * (dr)^2 \right] + n * (E_{elec} * k)$$

Indirect transmission with negligible data fusion cost would dissipate only the sum of *n* transmissions of distance *r*, and *n* receptions:

$$n * (E_{elec} * k + \in_{amp} * k * r^2) + n * (E_{elec} * k)$$

It is trivial to see that if each relay station also has a packet to send to the cluster-head that can be fused with the one it is relaying, then *it will always be more energy efficient for a node to transmit indirectly, so long as (1) it transmits the packet to a relay node that is closer to the cluster-head, and (2) it uses less power in this transmission than it would take to reach the cluster-head directly*. These are the two conditions that are satisfied by the route setup phase of HIT.

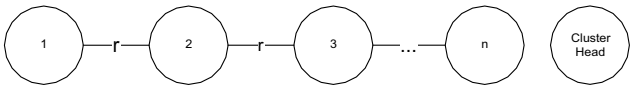

Figure 2. A linear network of *n* nodes that are distance *r* apart.

III.D. Parallel Transmissions

Delay is an important performance metric for a routing protocol, and one way to reduce delay is through the use of parallel transmissions. HIT attains a higher level of parallelism than both LEACH and PEGASIS. In LEACH and PEGASIS, the number of clusters limits the number of parallel transmissions. In contrast, HIT proposes an intelligent scheduling algorithm *that allows multiple, parallel indirect transmissions across multiple, adjacent clusters, with collision avoidance.*

III.E. Analysis of TDMA versus CSMA

HIT sets up a TDMA (Time-Division Multiple Access) schedule for its steady-state phase. This section justifies the choice of TDMA over CSMA (Carrier-Sensing Multiple Access).

CSMA wastes energy in an energy-constrained sensor network. The causes are collisions, overhearing, control packet overhead, and idle listening [21]. Collisions waste energy because data has to be resent. Overhearing is also a problem, since nodes expend energy on receiving and processing data not meant for them. CSMA also requires the transmission of control packets. The last problem is idle listening, when a node must spend energy listening for packets addressed to them. Idle listening consumes from 50% to 100% of the energy spent for receiving data [21].

The advantages of TDMA in a sensor network using a continuous data delivery model are no collisions, little overhead, and high energy-efficiency. Sensors can turn off their transmitters and receiver in between their assigned time slots. One disadvantage of TDMA is that when nodes die, it is difficult to change the TDMA schedule; HIT addresses this by periodically reorganising clusters. Another disadvantage is that TDMA requires time synchronization among nodes; HIT assumes it is given.

IV. Hit Protocol Description

HIT is broken up into rounds consisting of two periods: cluster setup, and a long steady state, where sensors continuously transmit data to the cluster-head. The data transmission phase is much

longer than the cluster setup phase to overcome the costs of setting up the cluster. Transmissions to the cluster-head are indirect, and follow the paths designated by the chains set up by our greedy algorithm. As messages are passed up the chain, they are fused together, until one final result is sent to the base station.

We make the following assumptions in our description of the HIT protocol:

1. Nodes are distributed randomly.
2. Nodes are able to communicate by CSMA using a known power-level that is agreed upon a-priori.
3. The application allows for two data items of fixed size s to be fused together, and the result of n fusions will be independent of n, and be no larger in size than a constant multiple of s.
4. Nodes are able to estimate distances (in the communication sense, not the spatial sense) to the originators of a particular CSMA signal by comparing their observed signal strength with the known power of transmission.[1]
5. Each sensor node has a unique ID, which is included in the header of each transmission it sends.

In the following, we use the terms *upstream* and *downstream* neighbors to describe nodes selected by the algorithm: when a node transmits upstream, it is relaying the message to a node that is closer to the cluster-head. Formal definitions and detailed schemes will be provided in section 4.3. Presented below is an overview of HIT protocol; detailed descriptions are given in the following subsections. Each round of the HIT protocol consists of the following phases:

1. Cluster-Head Election – One or more cluster-heads are elected.
2. Cluster-Head Advertisement – The cluster-heads broadcasts their status to the network.
3. Cluster Setup – Clusters and the upstream and downstream relationships are formed.
4. Blocking Set Computation – Each node computes its blocking set.
5. Route Setup – Sensors within a cluster form multi-hop routes to the cluster head.
6. TDMA Schedule Creation – A TDMA schedule is computed to allow for parallel transmissions.
7. Data Transmission – A long steady-state phase where sensed data is sent to the base station.

Phase 1 – Cluster Head Election

During this phase, one or more cluster-heads are elected: each cluster has one cluster-head. Below, we describe four election schemes for consideration; each of them has attributes that make it more suitable for certain applications. The first two schemes (Ia and Ib) are for a single cluster, and the latter two (IIa and IIb) are for multiple clusters. 'HIT' implies the use of Scheme Ia; likewise, a reference to 'HIT_m' – HIT with multiple clusters – implies Scheme IIa.

Election Scheme I(a): Single Cluster, Rotation

It is most energy-efficient to have only a single cluster-head: then only one node needs to make the large energy expenditure to transmit to the remote base station. If it is known a-priori that all nodes are able to communicate with the remote base station, then node ID can be used to decide the order of cluster-head rotation.

Election Scheme I(b): Single Cluster, Rotation, Additional Selection Criteria

In some networks it may be desirable to a elect a cluster head based on certain criteria, such as one with a certain level of connectivity to the remote base station, or with a power level that is sufficient to complete an entire steady-state phase of communication. In this case, cluster-head candidates must satisfy further constraints, in addition to node ID, before they are elected. When a node to act as the cluster-head has not met these criteria, then the subsequent node may respond after a time-out period.

Election Scheme II(a): Multiple Clusters, Random

The use of multiple clusters will reduce the gathering delay of the network, at the expense of the additional energy consumption required for multiple cluster-heads to transmit directly to the remote base station. Additionally, the use of multiple clusters will enable the network to continue operation under circumstances that have caused a temporary or permanent disconnection, but each independent cluster still has connectivity to the remote base station. In this scenario, an election scheme like that of LEACH [8] may be used. The desired percentage of cluster-heads, p, is selected a-priori. At the start of the round, each node i generates a random number $r \in [0,1]$ and compares it to a threshold $T(i)$ shown below, where G is the set of nodes that have not been cluster-heads in the last $1/p$ rounds. If r is less than $T(i)$, then the node becomes a cluster-head.

$$T(i) \{\text{if } (i \in G) \Rightarrow p / (1 - p^* (r \bmod (1/p)))$$
$$\text{else} \Rightarrow 0 \}$$

[1] Note that in multi-path environments, these estimates do not reflect spatial distances between nodes, but HIT does not rely on the ability of nodes to make spatial distance estimates.

Election Scheme II(b): Multiple Clusters, Random, Additional Selection Criteria

Like Scheme I(b), II(b) puts further restrictions on the set of candidate nodes by adding constraints to the threshold function:

$T(i)$ { if $(i \in G) \wedge (c > 50\text{dB}) \Rightarrow p / (1 - p^* (r \bmod (1/p)))$

else $\Rightarrow 0$ }

$T(i)$ { if $(i \in G) \wedge (b > 0.03\text{ J}) \Rightarrow p / (1 - p^* (r \bmod (1/p)))$

else $\Rightarrow 0$ }

Note that the above rules of election allow for rounds in which no cluster-head broadcasts an advertisement. For example, in the randomized election scheme there is a low probability that no cluster-heads will be elected. In this case and the case where the designated cluster head does not meet the selection criteria (in Schemes Ib and IIb), the network must come to a consensus after waiting some timeout period that the round must be aborted and the election process re-initiated. In a fully connected sensor network the minimum time-out period should be at least the round-trip propagation delay of the network.

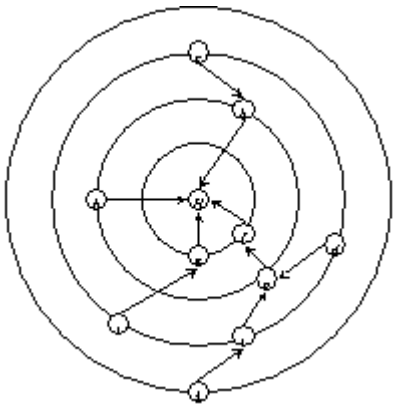

Figure 3. Consider node *k*, where *H* is the cluster-head. Since d(*k*, *H*) > d(*j*, *H*) AND d(*k*, *H*) > d(*k*, *j*), *j* may be an upstream of *k*. Note that *i* may also be an upstream of *k*. To resolve this, *k* chooses the closer node, *j*.

Phase 2 – Cluster Head Advertisement (CSMA)

In this phase, the elected cluster-heads for the current round broadcast their status at the fixed transmission power with the message:

<MsgType=Advertise, source-id = H>

During this time, non-cluster-heads must have their radio electronics on so they can listen for the advertisement. The non-cluster-heads then compute the distance to the cluster-head and save the value as *D(H, j)* where *j* is the node's ID.

Phase 3 – Cluster Setup (CSMA)

In this phase, one or more clusters are formed and the upstream and downstream relationship is created. At the beginning, each non-cluster-head broadcasts, using CSMA, the message:

<MsgType = Member, source-id = *j, D(H, j)*>

The message is broadcasted at the fixed transmission power. Then all non-cluster-heads listen for the membership broadcasts and use the observed signal strength to estimate the distance from their position to the sender of the broadcast.

In summary, after this phase has completed each node *i* will have the following information about every other node *j*, where *H* is the cluster-head of *j*:

1. *d(i, j)* – An estimated distance from node i to every other node j.
2. *d(j, H)* – An estimated distance from every other node j to j's cluster head H.

Next, each node computes its upstream neighbor. An upstream neighbor (which may be the cluster-head) is the next hop for a node during the data transmission phase. The upstream neighbor is responsible for receiving and fusing data from its downstream neighbors, and each non-cluster-head has one upstream neighbor. Upstream neighbors are calculated from the frame of reference of a node and its cluster-head. An upstream neighbor *u* of a node *i* with cluster-head *H*, is the *closest* node to *i* such that the following two conditions hold true:

1. $d(u, H) < d(i, H)$
2. $d(i, u) < d(i, H)$

The first condition is that cost of transmission to an upstream neighbor must be less than the cost of a transmission directly to the cluster-head. The second is that the upstream neighbor must be closer to the cluster-head than *i*.

Phase 4 – Route Setup (CSMA)

After each node has computed its upstream neighbor in the previous phase, all nodes broadcast this information at the fixed strength using a CSMA MAC, and include their estimate of the distance to their upstream neighbor in the message:

<MsgType = MyUpstream,
source-id = *j*, upstream-id = u_j, $d(j,u_j)$ >

This allows all other nodes to know (1) the upstream neighbor u_j of every other node *j*, and (2) $d(j,u_j)$, the distance from every node *j* to its up-

stream neighbor u_j. When a node hears a message from node *i* claiming that it is downstream from node *j*, it adds *i* to its list of downstream neighbors for *j, DOWN(j)*. This is important, because in the formation of the TDMA schedule, upstream nodes must wait for all of their downstream neighbors to transmit their sensor data so that all downstream sensor data is fused together before the result is passed up. Fig. 4 shows an example of the routes computed in HIT_m by the scheduling algorithm, with the desired percentage of cluster heads, *p*, set to 3% (with 100 nodes, this results in 3 clusters).

To conclude, at the end of this phase, each node records $d(j,u_j)$ and the set of downstream nodes *DOWN(j)* for all other nodes *j*.

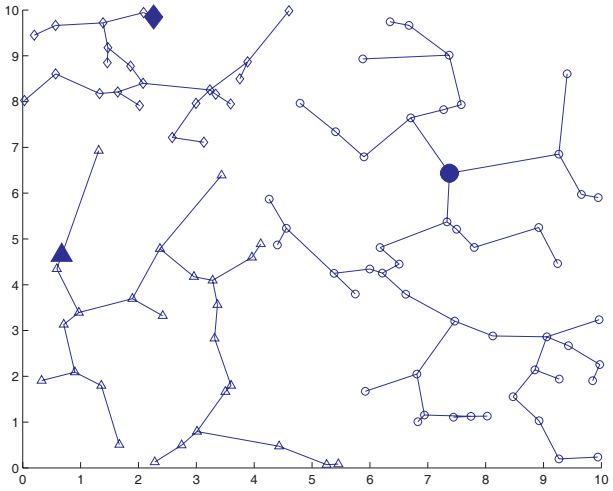

Figure 4. An example of routes computed by HIT_m. Each cluster is denoted by a particular shape, and cluster-heads are denoted by larger, solid versions of that shape.

Phase 5 – Blocking Set Computation (CSMA)

In this phase, each node computes the *blocking* set for its downstream neighbors using the information gathered in Phase 4. We say node *i* blocks node *j* if and only if

$$d(i, u_i) > d(i, u_j)$$

i.e., to reach its upstream neighbor, *i* must transmit at a power level that is strong enough to be heard at the upstream neighbor of *j*. This means that nodes *i* and *j* cannot simultaneously transmit to their respective upstream neighbors because a collision will be heard at u_j. Note that *i* blocks *j* does not necessarily imply that *j* blocks *i*.

At the end of this phase, every node broadcasts, at the fixed strength, one message, which contains the list of nodes that block its downstream neighbors:

<MsgType = Blockdown, node-id, blocklist>

Each node then merges the knowledge of the other nodes' downstream neighbors, which was obtained from the MyUpstream messages, to form a global BLOCKED-BY table that associates each node-id *j* with the set of nodes that cannot transmit while *j* is transmitting. Finally, the BLOCK table is built from the BLOCKED-BY table; it associates each node *j* with the set of nodes that block *j* from transmitting when any node in the set transmits. For every node *j*, each downstream neighbor of *j* blocks all other downstream neighbors of *j*. However, since these members of the block set can be implied from the list of downstream neighbors, they are not included in the Blockdown message.

Phase 6 – TDMA Schedule Setup

In this phase, echo node computes a TDMA schedule that allows close to the maximum number of nodes to communicate in parallel while maintaining a collision avoidance guarantee. Each node computes the same schedule independently and in parallel – under the assumption that the distance estimates are symmetric they will all compute the same schedule because they all have the same information. Before going into details of the scheduling algorithm it is useful to note that each node has obtained the following information during the previous five phases. Given a sensor node *i* where *H* is the cluster-head, *j* is any other node, and u_j is the upstream neighbor of node *j*, information stored in sensor node *i* includes:

1. d(j, H) – An estimated distance from node j to j's cluster head H.
2. d(i, j) – An estimated distance from i to j.
3. $d(j, u_j)$ – An estimated distance from j to j's upstream neighbor.
4. DOWN(j) – the set of downstream neighbors of j.
5. BLOCKED-BY(j) – the set of blocked nodes when node j is transmitting.
6. BLOCK(j) – the set of nodes that block j from transmitting.

To compute the TDMA transmission schedule, we begin by defining three sets of nodes:

1. READY – The set of sensors that have a message to send.
2. WAIT – The set of sensors that are ready, but need to wait due to blocking conditions.
3. $SEND_k$ – The set of nodes that may be sent at time slot k of the TDMA schedule.

Each TDMA time slot *k* is assigned to the $SEND_k$ set; the scheduling algorithm guarantees that up-

stream neighbors of the $SEND_k$ set never detect a collision when all nodes in $SEND_k$ transmit simultaneously. (It is possible for transmitting nodes to detect a collision, but the schedule guarantees that receiving nodes will not.) Furthermore, since we do not require the contents of *DOWN*(*j*) after the TDMA schedule has been generated, we modify it in-place. When the TDMA schedule has been generated, the *DOWN*(*j*) sets will all be empty.

Pseudo-code for the TDMA scheduling algorithm is given in Fig. 5. Initially, a node begins by adding all nodes with a packet to transmit to *READY*, and setting $k = 0$. The algorithm includes three loops. The first loop (Step 1) checks every node in *READY* and terminates when *READY* is empty. Inside this loop are two inner loops (Steps 2 and 3), one after the other. The first inner loop, Step 2, checks every node *j* in *READY,* and places *j* in the set *WAIT* if *DOWN(j)* is empty, i.e., all of *j*'s downstream nodes have been scheduled to send their data in an earlier slot of the current round.

```
Initially, set READY to be all the active sensors; set k=0;
1. while (READY is not empty) {
    2.  for (l = 0 ; l < size(READY) ; l++) {
    //a node in READY is placed in WAIT if all of its downstream nodes have been scheduled to transmit in earlier slots. //
        2a. m = READY[l]
        2b. if (DOWN(m) is empty) {
            2c. add m into WAIT
        }
    }
    3. for (l = 0 ; l < size(WAIT) ; l++) {
    //a node in WAIT is placed in SEND; all its blocking nodes are removed from WAIT to avoid collisions. //
        3a. m = WAIT[l];
        3b. remove m from WAIT
        3c. remove m from READY
        3d. add m into SEND_k
        3e. remove m from DOWN(u_m)
        3f. remove BLOCKED-BY(m) from WAIT
        3g. remove BLOCK(m) from WAIT
    }
// All the nodes in SEND_k may transmit simultaneously in time slot k of the TDMA schedule. //
k++;
 }
```

Figure 5. Pseudo-code denoting the algorithm each node runs to compute the TDMA schedule for parallel data collection.

Next, the algorithm enters the second inner loop (Step 3) that iterates through all the nodes in *WAIT*. Let $m = WAIT[l]$, the l^{th} item of *WAIT*. In the first iteration of Step 3a, it is safe for *m* to transmit in time slot *k* because $SEND_k$ is empty. However, we ensure that it will be safe to add the next *m* to $SEND_k$, by deleting BLOCKED-BY(*m*) and BLOCK(*m*) from *WAIT* in Steps 3f and 3g (these nodes will not be allowed to transmit in the same slot that node *m* transmits). Next, we remove *m* from *WAIT* and *READY* and add it to $SEND_k$ (Steps 3b and 3c). This inner loop terminates when *WAIT* is empty. Then, time slot *k* is incremented, and we repeat the outer loop again. An example of the execution of a TDMA schedule is given in Fig. 6.

The scheduling algorithm is designed such that each time we assign a node to *m* and add it to a $SEND_k$ set, three conditions hold true: (1) all downstream neighbors of *m* have been scheduled to send, i.e., been added to a previous *SEND* set (Steps 2b and 3e), (2) the transmissions of the nodes already in a $SEND_k$ set will not collide with the upstream neighbor of *m*, and similarly (3) the transmission of *m* will not collide with the upstream neighbors of any nodes already in the $SEND_k$ set (Steps 3f and 3g). Note that condition (1) also ensures that all data sensed in some round reaches the base station in the same round.

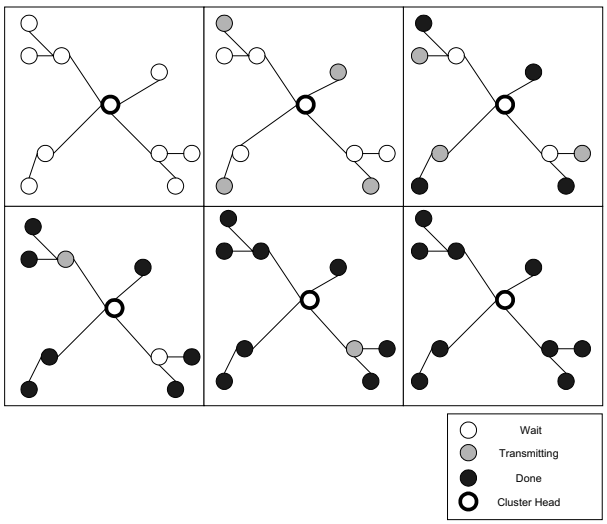

Figure 6. An example of data transmission using a TDMA schedule.

Phase 7 – Data Transmission (TDMA)

This phase is a long steady state phase where each node senses the environment and transmits to their upstream neighbor following the TDMA schedule created in the previous phase.

<MsgType=Data, source-id, destination-id, payload>

A node that receives data from its downstream nodes will fuse it together with its own, and transmit the result to its own upstream node. Eventually, data will reach the cluster head, which transmits its result directly to the remote base station. Note that because the TDMA schedule computed in the previous phase is followed, this phase does not require the use of a CDMA code within each cluster like the LEACH protocol does [8]. *Multiple, adjacent clusters can transmit in parallel with each other, and maintain collision avoidance.*

IV.A. Fault Tolerance

At the beginning of each HIT round, the set of active nodes are updated, and dead nodes (energy depleted nodes) are discarded. Additionally, HIT can be extended to deal with dead nodes within a single round. If a node dies during a round, HIT can use a timeout mechanism in each of Phases 1-5 to ensure that living nodes learn about dead nodes by their silence. During Phase 7, dead nodes may be removed from *READY* set after some silent period. After learning about dead nodes, several alternatives can be followed to fix a broken route (chain), as illustrated in Fig. 7.

Another problem that can occur is a network partition in which the partitions can still communicate with the base station, but not with each other. In this case, a single cluster-head will be able to serve only one partition. The problem may be easily fixed in the beginning of the next round when cluster-heads are re-elected (Phase 1).

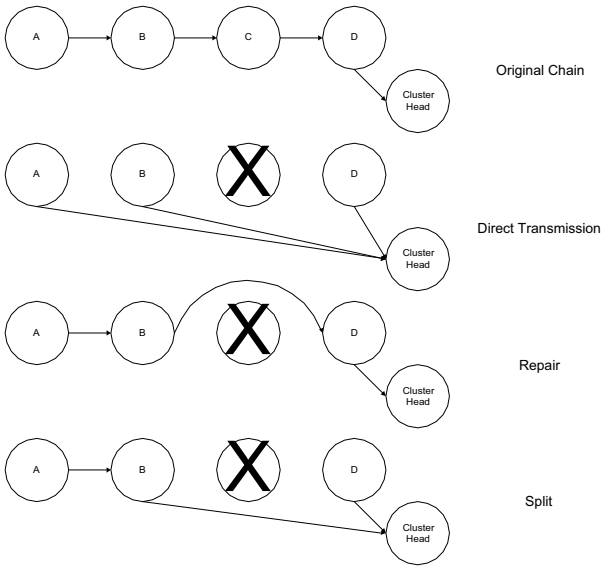

Figure 7. Fixing a route when node C dies within a round of HIT.

IV.B. Complexity Analysis

The major results of time and space complexity analysis are presented in this section. Detailed proofs are given in the Appendix.

IV.B.1. Time Complexity

The amount of time to execute one complete round of HIT is the time to complete all seven phases given in Sections 4.1 to 4.7. By carefully examining these seven phases, it can be seen that the TDMA Schedule Setup Phase (Phase 6) dominates the overall time. Thus, the analysis is focused on this phase and the algorithm given in Fig. 5.

Lemma 1

During the TDMA Setup Phase, collision occurs only among siblings, where siblings are defined as nodes sharing a common upstream neighbor node.

Proof Outline: It is shown that no node resides within the region that is closer than node *i* to the cluster-head and can hear node *i* transmitting to *u(i)* (the upstream neighbor of *i*), since it would contradict to the definition of *u(i)* – this node would be *i*'s upstream neighbor. Thus, the only nodes that can hear (and thus collide with) node *i* would be *i*'s siblings: those nodes that share the same upstream neighbor *u(i)*.□

Lemma 2

Let d be the height of the network (rooted at the cluster head). The average number of siblings per node is given below:

$$\frac{\int_1^d \frac{2h+1}{2h-1} - 1 dh}{d} = \frac{d + \ln d - d}{d} = \frac{\ln d}{d}$$

As d increases, the average number of siblings approaches zero:

$$\lim_{d \to \infty} \frac{\ln d}{d} = 0$$

Proof Outline: The above analysis assumes a random distribution of sensors within the area of a circle. The above theoretical result is supported by simulations of up to 2,500 sensors within a fixed network area for cluster-heads located at the network center, cluster-heads located randomly, sensors distributed randomly, and sensors distributed according to a uniform checkerboard pattern.

Theorem 3

The time complexity of the TDMA schedule setup phase, and thus of the HIT protocol, is O(n×log(n)), where n is the total number of sensors.

Proof Outline: The proof uses a simplified network where nodes have an upper bound on the number of downstream neighbors. That is, each node has s downstream neighbors; $s = \lceil S + 1 \rceil$ where S is the average number of siblings per node. It therefore provides an averaged worst-case analysis. From Lemma 1, on the average, the outer while loop (loop 1) of the TDMA scheduling algorithm (shown in Fig. 5) executes $D \times s$ times, where $D = log\ n$ is the average number of the depth of a network and the two inner for loops (2 and 3) each execute $2n$ times. Combining this with the result in Lemma 2, the total time complexity is $O(n \times log\ n)$.□

IV.B.2. Space Complexity

Theorem 4

The space complexity of HIT O(n).

Proof: The space complexity is essentially the space needed within each sensor node to execute the TDMA schedule setup phase (Phase 6), which consists of all the tables listed in Sec. 4.6. Note that each of these tables uses *O(n)* space. This is trivially true for Tables 1 – 3 listed. For the *DOWN* table it is true since each node has exactly one upstream neighbor; thus the total number of downstream neighbor of all the nodes is n. For the 5th and 6th tables, both the *BLOCK-BY* and *BLOCK* sets contain only siblings (Lemma 1) and the number of siblings is small (Lemma 2).

V. Performance Evaluation

This section discusses the testing methodology used in performance evaluation of HIT when compared to other protocols. It includes a description of the performance metrics and simulator environment, and a discussion of results.

V.A. Simulation Settings

The simulation network is made up of 100 nodes (the number of nodes are varied in later experiments) placed randomly on a 2 dimensional grid, and a base station (BS) some comparatively large distance from the center. All nodes are within the maximum transmission distance of each other, do not know the topology of the network and have the capability of varying the power of their radio transmitter.

In a round of communication, each sensor node has a data item to be sent to the distant base station. This data is propagated through the network following the HIT protocol. Data fusion occurs at each sensor. Thus, each sensor forwards only a single packet, regardless of how many it has received in a given communication round.

V.B. Simulation Parameters

The following simulation parameters remained constant for the duration of all the HIT simulations described in this paper. They differ slightly than what was used in earlier work [8, 13]. All nodes were bestowed with 20 J of initial energy, and the number of steady state loops for each round was fixed at 10,000. The data rate of the wireless network is 1 Mbit/s, and the coordinates for the base station are (l/2, -200), where l is the area side length. To more accurately reflect what we believe to be the average size of a sensor data item, a 50-bit packet size is used rather than the 2000-bit size [8, 13].

For LEACH and HIT_m we specified that 1% of the nodes would be elected cluster-heads ($p = 0.01$). Note that this value differs from what was previously determined to be optimal [1,8]. The reason is we found that having more than one cluster-head would unfairly drive up the average energy consumption per round. However, even when specifying a 1% cluster-head rule with 100 nodes results in some rounds where 2 or 3 are elected. In the simulations that varied the number of nodes, the value for p was kept constant at 1%. Thus, in simulations with 200 nodes the average number of nodes transmitting to the remote base station per round was approximately two, which is not optimal.

V.B.1. Performance Metrics

In this section, the HIT and HIT_m protocols are evaluated against LEACH, PEGASIS, and Direct Transmission. To rigorously evaluate their performance the following metrics were carefully chosen for comparison:

1. *Network Longevity* – This includes two metrics:
   a. *Number of rounds that occur before 100% of the nodes die.*
   b. *Number of living nodes as a function of time* – This metric shows how efficiently a protocol uses the aggregate energy to maximize network lifetime.
2. *Network Delay (Length of Rounds)* – This metric shows that the duration of a round is different, depending on the protocol. Thus, comparing network lifetime using rounds is misleading.
3. *Average energy dissipation* – Since nodes die in a discrete fashion, it is necessary to examine this concrete metric to resolve some possible performance ambiguities.

4. *Average energy dissipation × average delay product* – This metric measures the combination of two factors that are desired to be minimal [13]. Yet, these two metrics may become trade-offs as a protocol executes. It is therefore an important metric for determining the overall protocol performance.

V.C. Network Longevity

The *longevity* of the sensor network is the number of rounds until 100% of the nodes in the network have depleted their energy. Fig. 8 shows that HIT lasts about 1.05 times as long as PEGASIS and 1.44 times as long as LEACH, on an average sized area (10m x 10m) with 100 nodes and 50-bit packets. Lindsey *et al* found that PEGASIS lasted approximately 1.69 times as many rounds as LEACH did, using a 50m x 50m area, 100 nodes, 2000-bit packets, and the base station located at (25,150) [13]. In this experiment the base station was only 100m away from the nearest node and nodes had 0.25 J initial energy. By contrast, our simulation found that PEGASIS lasts approximately 1.37 times as many rounds as LEACH, which is believable given the differing simulation parameters used. With respect to longevity, HIT_m has very similar performance to LEACH.

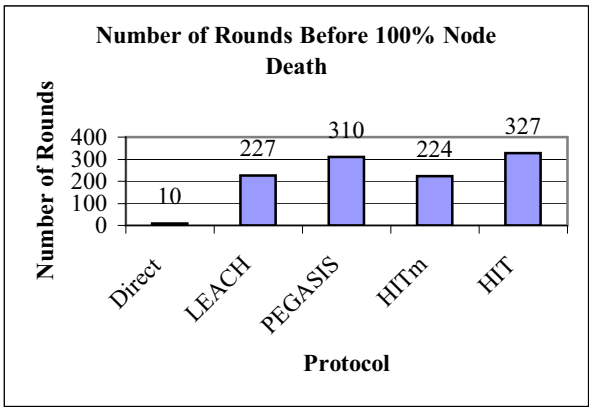

Figure 8. The number of rounds before all nodes died, using 100 nodes in a 10m x 10m area with 50-bit packets.

Next we explored the behavior of node death as a function of the round number. Fig. 9 shows the number of living nodes as a function of round number for each of the protocols. Protocols with more than one leader on average (LEACH and HIT_m) tend to die off more steeply than those with exactly one leader (PEGASIS and HIT). Direct Transmission dies so quickly that it has virtually no curvature at all.

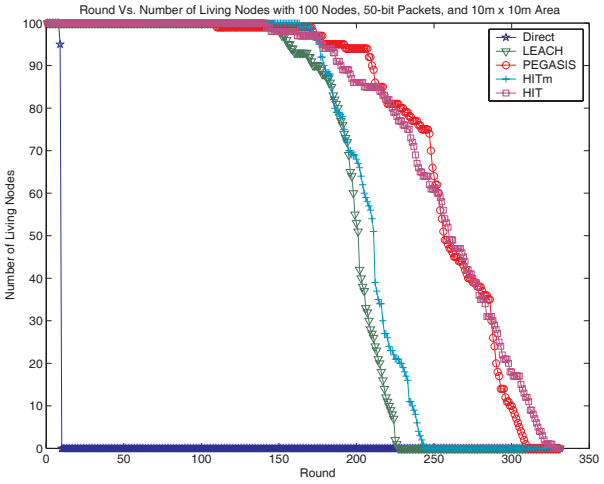

Figure 9. Round number vs. number of living nodes with 100 nodes, 50-bit packets, and in a 10m x 10m area.

When looking at the number of sensors alive at each round (Fig. 9) there is still some ambiguity over which protocol is the most energy-efficient. Although the last sensor in HIT dies later than the last sensor in PEGASIS there are certainly some rounds in which PEGASIS has more living sensors than HIT. The reason for this ambiguity is that sensors are chosen to transmit to the base station using a method that does not take power-level into account. This behavior introduces some variability in the number of sensors that die per round.

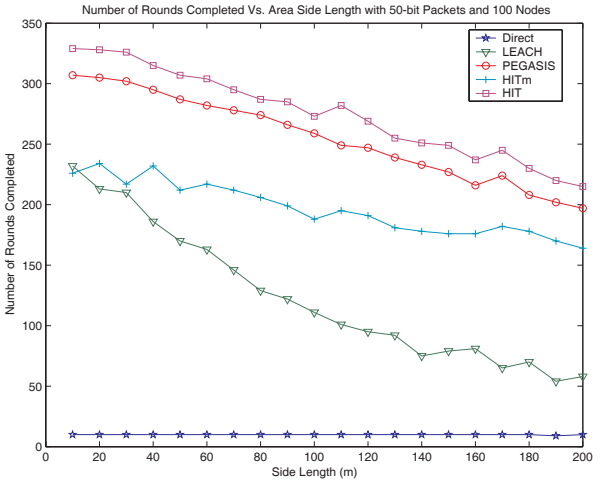

Figure 10. Number of rounds completed versus area side length, with 50-bit packets and 100 nodes.

Fig. 10 resolves some of this ambiguity by varying the area side length from 10 to 200 meters. As the figure shows, HIT and PEGASIS remain quite close, but HIT retains a fairly constant advantage. The number of rounds completed by HIT, HIT_m,

and PEGASIS descend at approximately the same rate as the area side length is increased. Although LEACH and HIT_m complete nearly the same number of rounds when the area side length is 10 meters, LEACH descends much more rapidly and for networks with an area side length larger than 10 meters, HIT_m is clearly preferable to LEACH. Direct expends so much energy to communicate with the base station that varying the area side length over the interval we chose did not result in a change in the number of rounds it completed.

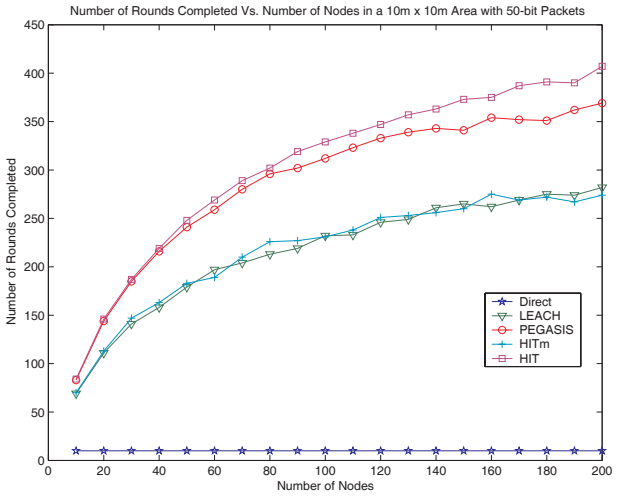

Figure 11. Number of rounds completed versus number of nodes in a 10m x 10m area with 50-bit packets.

Fig. 11 shows the number of rounds completed by each protocol as we varied the number of nodes in the simulation. HIT shows a slight advantage over PEGASIS when the number of nodes is small, and widens its lead as the network grows. LEACH and HIT_m appear to be quite close but keep in mind that the area side length was fixed at 10 meters. From the result shown in Fig. 10, one can expect that for area side lengths larger than 10 meters, HIT_m will maintain a constant advantage.

V.D. Length of Rounds (Network Delay)

Although HIT provides some improvement over PEGASIS in terms of the number of rounds before all nodes have depleted their energy, HIT provides significant savings over PEGASIS in terms of the length of time each round of data gathering consumes. Fig. 12 shows that on average, a round of data collection in HIT requires about 25% of the time required by PEGASIS or LEACH, measured in units of delay.

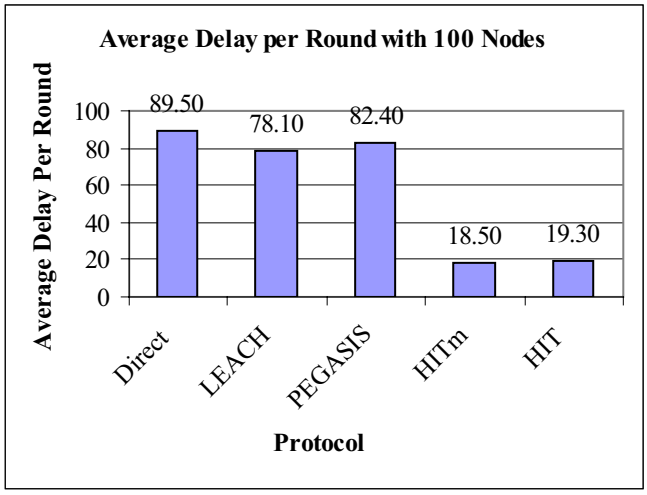

Figure 12. Average delay per round with 100 nodes, 50-bit packets, in a 10m x 10m area.

The average delay is computed over the course of a particular simulation and includes rounds in which some of the sensors have depleted their energy resources, and thus do not contribute data or incur units of delay. For this reason, with 100 nodes in the simulation, some rounds in Direct Transmission can be gathered in less than 100 units of delay – namely, the rounds in which there are less than 100 sensors with energy. This pulls down the average, and should be considered when our results are compared against other results. Fig. 13 shows the delay required for gathering data for each protocol in each round until all the nodes have depleted their power.

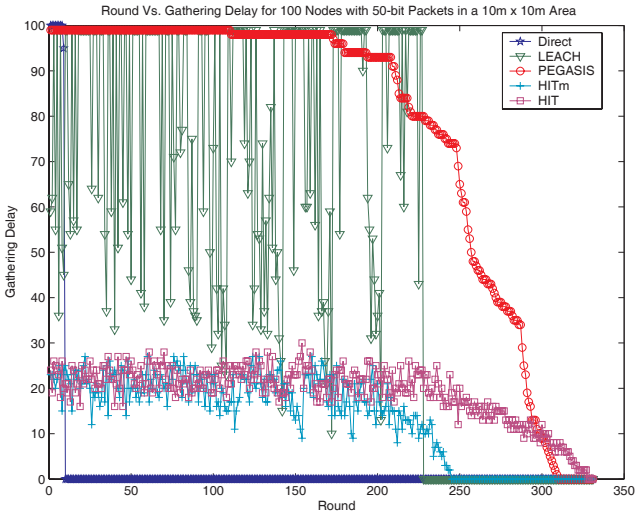

Figure 13. Round versus gathering delay for 100 nodes with 50-bit packets in a 10m x 10m area.

V.E. Energy Dissipation

Fig. 14 shows the average energy dissipation for each protocol on a logarithmic scale as the area side length was varied from 10 to 200 meters. HIT and PEGASIS are quite close in average energy dissipation. This fact, coupled with the significant energy depletion that occurs when a node transmits to the base station explains the intertwining effect shown between the traces of HIT and PEGASIS in Fig. 9.

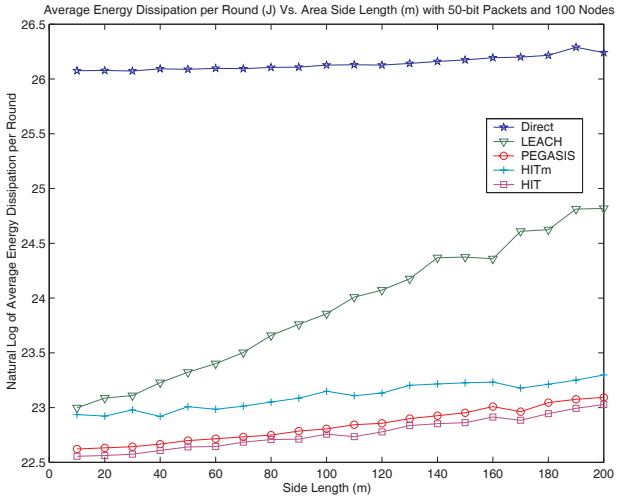

Figure 14. Average energy dissipation per round by protocol versus area side length with 50-bit packets and 100 nodes.

Although it is not obvious from Fig. 14 because the scale is skewed so harshly by Direct Transmission's comparatively poor performance, the gap between PEGASIS and HIT is growing as the area side length is increased. HIT's average energy dissipation grows more slowly than that of PEGASIS as the area side length is increased, which implies that for large areas, the difference between the two protocols will be significant.

Fig. 15 shows the average energy dissipation per round as the number of nodes is varied from 10 to 200. HIT's average energy dissipation grows more slowly than PEGASIS's as the number of nodes is increased. Thus, for large numbers of nodes, the difference between the two protocols will be significant.

V.F. Energy x Delay

The final metric considered is a combination of the energy dissipation and delay and most accurately captures the true performance of each protocol. The delay for each protocol decreases as nodes die off and no longer incur units of delay. To compute the average energy × delay product, the energy dissipated in each round is multiplied with the delay consumed by that round. These values are then summed, and divided by the number of rounds the protocol completed before all nodes had depleted their energy; the results are shown in Fig. 16.

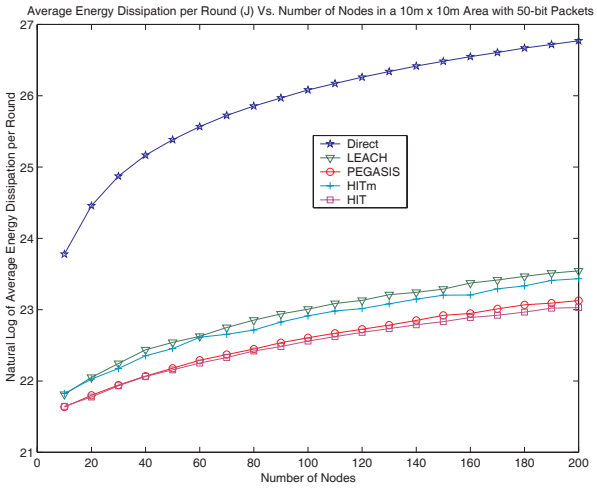

Figure 15. Average energy dissipation per round versus number of nodes in a 10m x 10m area with 50-bit packets.

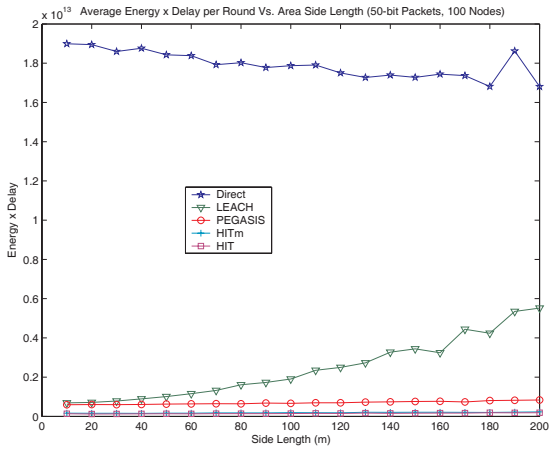

Figure 16. Average energy x delay per round versus area side length with 50-bit packets and 100 nodes.

Fig. 16 is also skewed considerably by Direct Transmission's comparatively poor performance. In order to show a more informative figure, we regenerated the exact same plot, below, in Fig. 17, but suppressed the trace of Direct Transmission. Although this helped somewhat, perhaps it is not obvious that average energy × delay product of PEGASIS does indeed grow more quickly than that of HIT. As the number of nodes is varied, however, the difference between PEGASIS and HIT grows at an accelerating rate, as is shown in Fig. 18, and more clearly in Fig. 19.

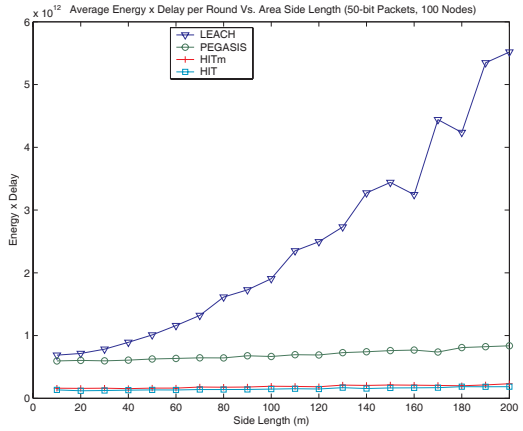

Figure 17. Average energy x delay per round versus area side length with 50-bit packets and 100 nodes. This is the same as Fig. 16, except the Direct protocol is not shown, which gives the other protocols finer resolution.

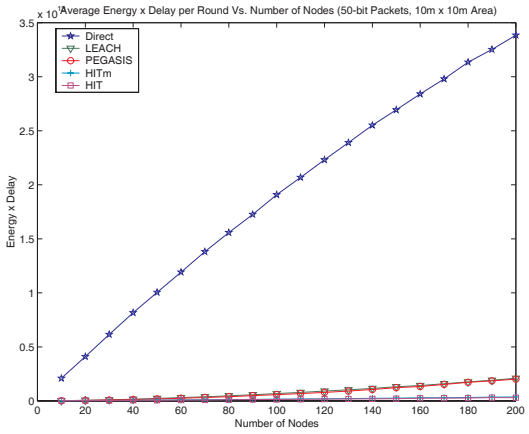

Figure 18. Average energy x delay per round versus number of nodes with 50-bit packets in a 10m x 10m area.

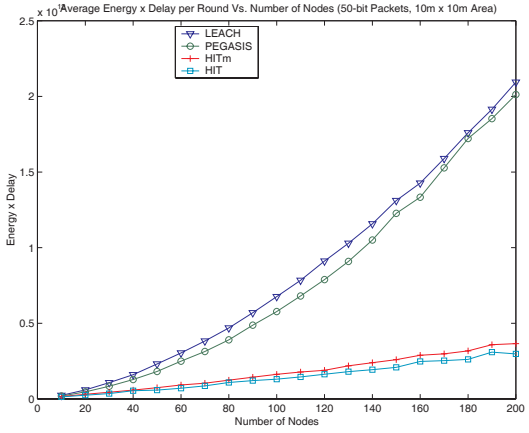

Figure 19. Average energy x delay per round versus number of nodes with 50-bit packets in a 10m x 10m area. This is the same as Fig. 18, except the Direct protocol is not shown, which gives the other protocols finer resolution.

VI. Security

This section discusses security considerations of sensor networks in general, and of HIT in particular. Sec. 6.1 describes types of attacks that have been shown to be effective against sensor network protocols. Sec. 6.2 discusses specific security issues of HIT.

VI.A. Attacks to Wireless Sensor Networks

Due to the broadcast medium, wireless networks are inherently insecure. Wireless sensor networks are vulnerable to eavesdropping, malicious hosts masquerading as sensor nodes and injecting falsified data into the stream, and denial-of-service attacks in which adversaries jam the radio communication channels. Furthermore, wireless sensor networks running higher-level protocols that rely on the cooperation of nodes in the system are vulnerable to attacks by malicious nodes that subtly or obviously refuse to cooperate. A distinction may be made between the former class of attacks, which try to manipulate data directly, and the later, which affect the underlying routing topology [10].

Attacks that are most relevant to HIT are network layer attacks. These attacks may be categorized into seven groups, which are briefly described as follows:

1.*Spoofed, altered, or replayed routing information*: Adversaries may be able to create routing loops, attract or repel network traffic, extend or shorten source routes, generate false error messages, partition the network, or increase end-to-end latency.

2.*Selective forwarding*: In networks that assume the participants can be trusted to relay messages, adversaries may choose to selectively drop them, to prevent certain messages from propagating further in the network.

3.*Sinkhole attacks:* This attack involves coercing neighboring nodes to relay packets through them, enabling further attacks such as selective forwarding.

4.*Sybil attacks:* A single node presents multiple identities to other nodes in the network.

5.*Wormhole attacks*: An adversary tunnels messages from one side of the network to another, through a low latency communications channel, and replays them.

6.*HELLO flood attacks*: An adversary abuses the assumption of many sensor network protocols that the reception of a certain message type indicates that a neighbour node is nearby – an adversary with

a powerful radio might convince all of the nodes in a network that it is 'nearby,' and has a high quality route to the base station. Those nodes sufficiently far away from the adversary would be sending packets into oblivion, needlessly wasting energy.

7.*Acknowledgement spoofing*: An adversary may try to convince the network that a dead node is alive, or that a weak link is strong, by fabricating link layer acknowledgements that appear to originate from another node.

VI.B. Security Considerations for HIT

This section first presents a general description of vulnerabilities and possible attack scenarios of HIT. Next, two specific attacks and their detections and preventions are carefully described. The section is concluded by a brief discussion of graceful degradation.

VI.B.1. Vulnerabilities of HIT

HIT, as described so far, is vulnerable to sinkholes, selective forwarding, and Sybil attacks; all other forms of attacks mentioned will have little or no effect. Furthermore, selective forwarding and Sybil attacks can easily be detected in a HIT network by implementing a few routines that defensively verify the correctness of the protocol, as outlined below. For example, HIT requires that each node broadcast one packet per data collection round. It is thus able to detect a packet that is not sent, or the presence of a previously unknown node. In the text that follows, the focus is on minimizing the effect of a sinkhole attack against HIT.

Attack Scenarios in a HIT Network

There are two distinct classes of attacks that can be mounted against a wireless sensor network:

1. *Malicious Node Attacks*: The first case is a network into which a few malicious nodes have been randomly placed, or some of the original participants have been electronically hijacked: the attackers have hardware that is similar or identical to that of the original nodes, and is thus similarly limited.

2. *Laptop-class Attackers*: This class assumes that attackers have access to more powerful devices.

Since HIT requires variable power transmitters and does not assume that the known, fixed CSMA transmission power level is the maximum at which radios in the system can transmit, the most significant difference between these two classes in the context of HIT is that multiple laptop-class attackers may be able to coordinate their attack via out-of-band networks. This capability will be leveraged in an attack scenario in a future work.

Focus now on the case in which an attacker has infiltrated the area occupied by a HIT network with a laptop. The attacker's goal is to avoid detection while poisoning the information sent from the sensor network to the remote base station. Note that a hardened version of HIT (described more in Sec. 6.2.5), which verifies the correctness of the protocol at each node as it executes, would immediately detect an intruder if two data items are sent from the same node ID in the same round. Thus, in the following discussion of two specific attacks and preventions, it is assumed that an attacker physically captures a node, turns it off, and then immediately begins masquerading as the attacked node, before any omitted packet transmissions occur (these would be detected by other nodes).

VI.B.2. Cluster Head Attacks and Detection

Recall that our attacker's goal is to avoid detection while poisoning the information sent to the remote base station. The simplest way to accomplish this goal is to become a cluster-head. Note, however, in election scheme I, the election is determined a-priori by the node ID. The self-checking HIT implementation ensures that any cluster-head advertisements that are heard come from an appropriate source. Thus, an attacker against a network using election Scheme I is forced to wait until the order of node IDs designates the captured node as being elected. Subsequently, the attacker will be cluster-head for one steady-state phase, but will be forced to relinquish its position for n-1 steady-state rounds after its round is over. In election schemes IIa and IIb, it is more difficult to detect such attacks. Since the processes in IIa and IIb are random it is acceptable, though highly improbable, for a node to be elected in multiple, sequential rounds. In this case, a self-checking HIT implementation must rely on a threshold of normalcy for the number of times in a window of rounds that a node can be cluster-head before triggering a recourse action.

VI.B.3. Upstream Neighbor Attacks and Detection

During rounds when the threat of detection prevents an attacker from being a cluster-head, the attacker can divert traffic by advertising false information in the Cluster Set Up Phase in HIT. In the following, the case of an upstream neighbor attack is described.

Recall from the Cluster Set Up Phase (Phase 3, Sec. 4.3) that an upstream neighbor $u(i)$ of a node i is the closest node to i such that it costs less for i to transmit to $u(i)$ than to transmit directly to the clus-

ter-head, and that *u(i)* is closer to the cluster-head than *i*. An attacker could falsely advertise that it heard the cluster-head advertisement at a very high power. The recipients of that message would think that the malicious node is close to the cluster-head, and would give it preference as an upstream neighbor. Furthermore, since HIT relies on a known, fixed power for computing communication distances, a malicious node could easily make other nodes think that it is closer by simply broadcasting at a power that is higher than this pre-determined strength.

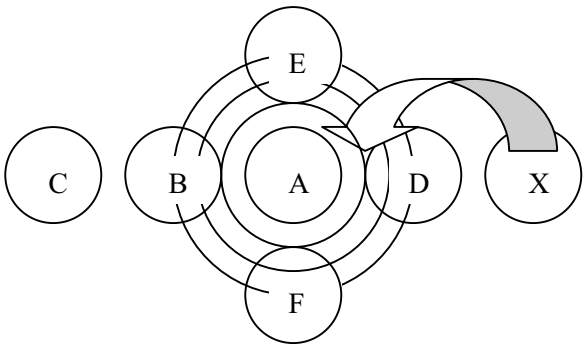

Figure 20. Malicious node X tries to convince node D that D is downstream from X, even though X is farther away from the cluster head, A.

An Example of Upstream Neighbor Attack

Consider the following scenario, in which X is a malicious node, A, B, C, D, E, and F are nodes following the protocol, and A is the cluster-head for the current round. Due to random placement, B is 5 m to the left of A, C is 10 m to the left of A, D is 5 m to the right of A, and X is 10 m to the right of A. Node A advertises itself cluster-head. All nodes receive this message, and then send a response that indicates they belong to A's cluster and contains the power they heard the advertisement message at. The malicious node X sends a message that indicates it is only 1 cm from A, at a high power that makes even C hear the signal at the known, fixed power. Node C must simply infer that X is 1 cm from A and right next to C since C has no reason to believe that X is violating the protocol. But all the other nodes heard X at a power that is much larger than the known, fixed power. The self-checking version of HIT protocol would detect that the malicious node X had sent a signal over the power threshold value and report X as an attacker.

X can certainly trick D into being that D is a downstream neighbor of X, without triggering the power threshold. X can do this by broadcasting the same false message at a power level that will be detected by D as slightly less than the known, fixed power. This works because during the Cluster Setup Phase (Phase 3, Sec. 4.3), each node collects *only* information about its distance to its cluster head and to all other nodes.

Detection of Upstream Neighbor Attacks

In addition to using a threshold signal power, it is possible to minimize the ability of malicious nodes to recruit downstream in the aforementioned manner by *adding a phase prior to the route set up, in which nodes exchange information that they have about other nodes*. After the exchange, they cross validate the information, and look for nodes that have advertised positioning information that is impossible or unlikely. For example, if nodes E, F, and D were to cross validate their opinions of where X is, according to the false information X broadcasted they would discover that they were in mutual disagreement. A full information exchange would require each node to broadcast *n* pieces of information, resulting in n^2 transmissions during this additional phase. To avoid this additional overhead, we propose that at each round, each node randomly chooses *k* other nodes with which to cross-validate their information. Thus, one can expect to detect attacks based on the premise of broadcasting false information within *N/k* rounds.

VI.B.4. Graceful Degradation

It is most desirable for the performance of a sensor network to degrade gracefully as nodes are compromised – i.e., a network with 10% of the nodes compromised should still operate at 90% efficiency. In the discussion presented so far, a self-checking version HIT protocol has been outlined under the assumption that the network has some mechanism of notifying all nodes in the network when a breach has occurred, and together, the nodes act on this trigger by simultaneously engaging in some evasive action, such as network shutdown, or the renegotiation of encryption keys, transmission frequencies, or CDMA codes.

VII. Sensor Network Applications: Present and Future

This section first presents an overview of some noteworthy current-day sensor network deployments; some of them are quite different from what HIT is targeted for: systems that will be deployed approximately 3 to 5 years in the future. An important potential application of the HIT protocol, data gathering for sensors in bioelectric interfaces to

computers, is then described.

VII.A. Present Deployments of Sensor Networks

In March of 2001, UC Berkeley and MLB Co. demonstrated their ability to deploy a sensor network from an unmanned aerial vehicle (UAV) onto a road, and subsequently use the sensor network to detect and track vehicles moving through it [24]. The network, which was composed of only six sensors, was dispersed over a small road, about 30 meters across. The sensors landed about 5 meters apart from each other, resulting in a full connectivity within the sensor network, which was leveraged to time-synchronize the sensors. Connectivity with the remote monitoring station was episodic and accomplished by way of the overhead UAV flying back and forth between the drop target and the base camp. The sensors stored logs of their measurements until such time as they could transmit them to the UAV. No power-management protocols were used to disseminate data to the UAV, and no data fusion was done to reduce the size of transmitted results: each node transmitted its sensor readings directly to the UAV.

Mainwaring *et al* [14] deployed 32 sensor nodes in a network on a small island off the coast of Maine to gather environmental data that they hoped would help answer research questions about seabird nesting patterns. Some of the sensors were placed in underground burrows, limiting the transmission range of their radios. The network architecture was as follows. The sensors were divided into loosely connected 'patches' of about five nodes which communicated with a gateway node either directly or indirectly through other nodes. Together these gateway nodes formed a 'transit network,' which served to forward data to the base station. While the sensor nodes themselves relied on only batteries as a power source, the gateway nodes were fully powered by solar panels, and thus were less power-constrained. Network data aggregation was performed in appropriate cases – for example, reporting only the average temperature across a region – but no robust energy-efficient data gathering algorithm was utilized.

Kottapalli *et al* proposed a sensor network architecture and protocol for structural monitoring, and implemented a small experimental instance of it [11]. Their network architecture consists of two tiers that communicate on separate frequencies: sensor units communicate with local site masters using the 915 MHz band, and local site masters communicate with the central site master on the 2.4 GHz band. The authors' attention to energy efficiency was directed primarily at achieving a sensor network lifetime which would exceed the service cycle period. Their network architecture is static, and no attempt was made to alleviate the non-uniform energy loss due to local site masters relaying information from the sensor units to the central site master. The architecture was designed to support approximately 10 sensor units per local site master.

These three systems are representative of the current state of the art in terms of sensor network deployments. Although they certainly demonstrate admirable feats, it should be noted that they are fairly simplistic in comparison to what the future holds: one utilized only six sensors, and the other two have manually placed sensors and static network topologies. Furthermore, all three examples used nodes that are fairly large, at least one cubic inch in size.

VII.B. Future Application of HIT

The design of the HIT protocol looks approximately 3 to 5 years into the future, when sensor network deployments will be significantly different – the nodes will be smaller and so numerous that manual placement will be virtually impossible. Instead, the sensors will be dispersed randomly and will rely on self-organizing communication protocols to formulate an energy-efficient data-gathering scheme.

VII.B.1. Bioelectric Computer Interfaces

One specific application HIT is targeted for is data gathering from sensors used for bioelectric computer interfaces. In the Neuro-Engineering Laboratory at NASA Ames Research Center where the first author works, researchers are building software and hardware systems for deciphering electro-myogram (EMG) and electro-encephalogram (EEG) signals, and using them to control computers [23]. Similar work is being done in Japan at Hokkaido University [15], Hiroshima University, and the National Institute of Advanced Industrial Science and Technology in Tsukuba [5]. Currently, using the research systems involves coupling anywhere from two to 256 electrodes to a human body, and reading signals from it as muscles are flexed (EMG) or thoughts are produced (EEG). The electrodes are wired to an amplifier that feeds into an analog-to-digital acquisition system. Once digitised, the signals are processed in a variety of ways, and conclusions are drawn as to how the human wishes to manipulate a computer interface. A system that requires 256 wires running off of a hu-

man arm or head to a computer is too cumbersome for most applications, and thus the researchers wish to move to a wireless platform as soon as it is feasible.

VII.B.2. HIT for EMG Sensing in Bioelectric Computer Interfaces

Based on the above discussions, the desirable features of a sensor network for EMG sensing may be outlined as follows:

Requirements

1. *Numerous, tiny sensors*: It seems that more sensors are better – and, more importantly, it is desirable for the sensing area covered by a single sensor to be as small as possible. Thus, as the sensors develop and decrease in size, it is conceivable that these systems will have to organize tens of thousands of different sensing nodes.

2. *Data fusion*: Rather than broadcasting raw sensor data to the base station, it is desirable for the sensor network to first process these data, and extract information that implies certain muscle groups have fired. At spatial resolutions that are high enough to detect individual motor unit action potentials, EMG recordings from multiple locations on the surface of the skin can be regarded analogously to audio recordings of a cocktail party using multiple microphones, for which the beam-forming model of data fusion applies. The details of applying the beam-forming model to EMG data will be dissected in a future work.

3. *High energy efficiency*: We expect that each sensor will be installed with a tiny battery that must store enough power to last several years.

4. *Continuous data model*: In these recordings, data acquisition must be continuous.

Conditions

1. *Network topology*: For EMG sensing, the distance between sensors will be on the order of millimetres, and the remote base station will be somewhere in the room. Thus, the sensor network will be both fully connected internally, and with the remote base station.

2. *Imprecise placement of sensors*: It is envisioned that the sensors will be placed uniformly, but imprecisely, to keep production costs low.

Currently there is no available technology that can fulfil these requirements while operating under these conditions. Wireless sensors with data fusion capabilities and enough power for continuous operation are not yet small enough to achieve the spatial resolution required, and the wiring of a conventional sensor network over an area this small with a large number of nodes would be an impossible task. However, we expect that hardware to meet these requirements will soon be available, and believe that a promising step towards organizing and gathering data from sensors in this and other similar biomedical sensing applications would be to augment HIT with an application-specific data-fusion model.

VIII. Conclusion

We have described HIT (and HIT_m for multiple clusters), a hybrid clustering and indirect transmission scheme for micro sensor networks. The novel feature of parallel, indirect transmissions and the complete protocol including phases of clustering, routing, and scheduling have been presented in detail. Its complexity was analysed and the fault tolerance mechanisms were presented. Performance evaluation has shown that HIT provides energy savings over LEACH, PEGASIS, and Direct Transmission for small areas and small numbers of nodes. This advantage becomes more significant for large areas or large number of nodes. HIT also greatly reduces the delay required to gather data from all sensors in a network by utilizing parallel, indirect transmissions, which requires neither the remote base station to compute the data gathering schedule nor sensor nodes with CDMA capability. This paper also described security issues and a potential application of HIT. Enhancing HIT to address these issues, and working closely with biomedical engineers on tailoring HIT for their specific applications would be the major focuses of future work. Others include 1) enhancements to HIT by integrating fault tolerant mechanisms, 2) incorporating some optimal clustering algorithm [1, 6, 22] into the protocol, and 3) applying specialized MAC protocols for sensor networks such as SMAC for event-driven data model [21].

Acknowledgment

The authors greatly appreciate the many helpful discussions with Dr. Kevin Wheeler of the Neuro-Engineering Laboratory at NASA Ames Research Center on potential applications of wireless sensor networks for bioelectric computer interfaces.

References

[1] S. Bandyopadhyay and E.J. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for

Wireless Sensor Networks," *Proc. of IEEE INFOCOM 2002*, New York, NY, June, 2003.

[2] S. Basagni, "Distributed Clustering for Ad Hoc Networks," *Proc. of International Symposium on Parallel Architectures*, *Algorithms and Networks (I-SPAN '99)*, pp. 310-315, Fremantle, Australia, June, 1999.

[3] M. Chatterjee, S.K. Das, and D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks," *Journal of Cluster Computing, Special issue on Mobile Ad hoc Networking*, Kluwer, No. 5, pp. 193-204, 2002.

[4] B. Deb, S. Bhatnagar and B. Nath, ReInForM: "Reliable Information Forwarding using multiple Paths in Sensor Networks," *DCS Technical Report DCS-TR-495*, Rutgers University, March 2002.

[5] O. Fukuda, J. Arita, and T. Tsuji, "An EMG-Controlled Omnidirectional Pointing Device Using a HMM-based Neural Network," IEEE IJCNN, Portland, Oregon, USA, July 21-25, 2003.

[6] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, "Optimal Energy Aware Clustering in Sensor Networks," *Sensors Magazine*, MDPI, Issue 1, pp 258-269, January, 2002.

[7] W. Heinzelman, "Application-Specific Protocol Architectures for Wireless Networks," Ph.D. thesis, Massachusetts Institute of Technology, 2000.

[8] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *IEEE Proc. Of the Hawaii International Conf. on System Sciences*, January, 2000.

[9] W. Heinzelman, J. Kulik, and H. Balakrishnan, " Adaptive protocols for information dissemination in wireless sensor networks," *Proc. of MOBICOM 1999*, pp. 174-185, Seattle, WA, August, 1999.

[10] C. Karlof and D. Wagner, "Secure Routing in Sensor Networks: Attacks and Countermeasures," Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA), Anchorage, AK, USA, May 11, 2003.

[11] V. Kottapalli, A. Kiremidjian, J. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei, "Two-tiered wireless sensor network architecture for structural health monitoring," SPIE 10th Annual International Symposium on Smart Structures and Materials, San Diego, CA, USA, March 2-6, 2003.

[12] Q. Li, J. Aslam, and D. Rus, "Hierarchical Power aware Routing in Sensor Networks," *Proc. of the Discrete Mathematics and Computer Science Workshop on Pervasive Networking (DIMACS '01),* Piscataway, NJ, May, 2001.

[13] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data Gathering Algorithms in Sensor Networks Using Energy Metrics," *IEEE Transactions on Parallel and Distributed Systems*, September 2002, pp. 924-935.

[14] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. "Wireless Sensor Networks for Habitat Monitoring," *ACM International Workshop on Sensor Networks and Applications (WSNA '02)*, Atlanta, GA, September, 2002.

[15] D. Nishikawa, Y. Ishikawa, W. Yu, M. Maruishi, I. Watanabe, H. Yokoi, Y. Mano, Y. Kakazu, "On-line Learning Based EMG Prosthetic Hand," The XIII Congress of International Society of Electrophysiology and Kinesiology (ISEK2000), Sapporo, Japan, June 2000.

[16] V. Rodoplu and T. Meng, "Minimum energy mobile wireless networks," IEEE JSAC, 17(8), pp. 1333-1344, 1999.

[17] C. Savarese, J. Rabay and K. Langendoen, "Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks," *USENIX Technical Annual Conference*, Monterey, CA, June 2002.

[18] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models," *ACM Mobile Computing and Communications Review* (MC2R '02), 2002.

[19] B. Warneke, M. Last, B. Liebowitz, K.S.J. Pister, "Smart Dust: Communicating with a Cubic-Millimeter Computer," *Computer Magazine*, IEEE Computer Society, Vol. 34, No. 1, pp 44-51, January, 2001.

[20] M.D. Yarvis, W.S. Conner, L. Krishnamurthy, A. Mainwaring, J. Chhabra and B. Elliott. "Real-World Experiences with an Interactive Ad Hoc Sensor Network," *IEEE International Conference on Parallel Processing Workshop 2002 (ICPPW 2002),* Vancouver, BC, August 2002.

[21] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," In Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2002, New York, NY, June, 2003.

[22] M. Younis, M. Youssef and K. Arisha, "Energy-Aware Routing in Cluster-Based Sensor

Networks," *IEEE/AMC Modeling Analysis and Simulation of Computer and Telecommnunications Systems MASCOTS '02)*, Fort Worth, TX, October 2002.

[23] http://ic.arc.nasa.gov/publications/pdf/2000-0284.pdf

[24] http://robotics.eecs.berkeley.edu/~pister/29Palms0103/

Appendix: Detailed Proofs of Complexity Analysis

This appendix section provides detailed proofs for our time complexity analysis of HIT.

A.1. Proof of Lemma 1 – Collisions Can Occur Only with Siblings

The condition for choosing upstream neighbors insures that in the average case, the transmissions of nodes sending data to their respective upstream neighbor only collide with those of their siblings. Examine Fig. 21, which shows a node *i* with its upstream neighbour *u*, and cluster-head *H*. Two large circles represent the transmission regions from *i* to *H (*called *circle I)*, and vice versa (called *circle H*), while the small circle represents the transmission region from *I* to *u(i)* (where a collision could occur when *I* transmits to *u(i)),* called *circle u.* Areas *X* and *Y* represent the intersections of *circles u and I*, and *circles u and H*, respectively. We can reasonably assume that no other nodes share the same relative distance from node *i* to *u* because the probability is negligible in a random distribution. Thus, region *Y* must be empty. To prove this, assume that *Y* is not empty. Then there exists a node *j* in *Y* such that d(*i, j*) < d(*i, u*). This is a contradiction since *i* would have chosen *j* instead of *u* as its upstream neighbor. Therefore, *Y* must be empty.

Region X must, on average, be sparsely populated. Let *x* be the number of nodes in X. The distribution of *x* is Poisson with mean and variance λ, and thus the probability of finding exactly *x* nodes in region X is:

$$p(x) = \lambda^x e^{-\lambda} / x!$$

Let ρ be the density of nodes per unit area and *A* be the area of X. λ can be shown to be equal to ρA. p(*x*) then becomes:

$$p(x) = (\rho A)^x e^{-\rho A} / x!$$

If *x*=0 then the probability *A* is empty is:

$$p(0) = e^{-\rho A} = \exp(-\rho A)$$

Assume that the area of *A* is πr^2 and let:

$$r = 1/2\sqrt{\rho}$$

,where r is the average distance between nearest neighbors.

$$p(0) = \exp(-\rho A) = \exp(-\rho\, \pi r^2)$$

$$= \exp(-\rho\, \pi\, (1/(2 / \sqrt{\rho}))^2)$$

$$= \exp(-\pi/4)$$

$$= .46$$

The expected value of *x* is:

$$E[x] = \lambda = \rho A = \pi/4$$

$$= .78$$

Since X is sparsely populated, any node *j* in X will with high probability select *i* as its upstream neighbour. This is because *j* will most likely be the only node in *i* and *d(i,j) < d(i,u) = r* where *r* is the average distance between nearest neighbors.

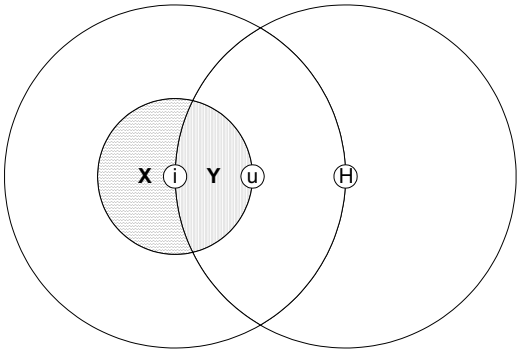

Figure 21. When *i* transmits to its upstream neighbor *u*, any node in the region X or Y will hear the transmission.

A.2. Proof of Lemma 2 – The Average Number of Siblings is Small

Through simulation, we have shown that the average number of siblings per node in a HIT network is small. Table 2 and Fig. 22 show the average number of siblings using random and uniform distributions, and using centered and random cluster-heads. The uniform distribution used in the simulation was a checkerboard pattern. Note that the HIT scheduling algorithm, when applied to a uniform checkerboard distribution, is equivalent to finding the minimum Manhattan distance from each node to the cluster-head. The Manhattan distance is the distance between two points while travelling parallel to the axes.

The data shows that uniform distributions produce HIT networks with higher than average numbers of siblings than those with random distributions. As the number of nodes in the network increases, the average number of siblings decreases. Cluster-heads near the center of the network have a higher average number of siblings than those on the boundaries

of the network. The reason for this is that a network with a cluster-head near the center has less depth than a network with a cluster-head at the boundary.

N	Random Distribution		Uniform Distribution (Checkerboard Pattern)	
	Centered Head	Random Head	Centered Head	Random Head
9	0.598	0.733	0.943	1.628
25	0.558	0.648	0.932	1.120
49	0.547	0.581	0.928	0.960
100	0.536	0.558	0.924	0.958
400	0.486	0.496	0.659	0.790
1000	0.467	0.476	0.651	0.763
2500	0.460	0.464	0.647	0.693

Table 2. The average number of siblings per node in HIT networks with random and uniform node distributions. A 'centered head' is a cluster-head nearest to the center of the network.

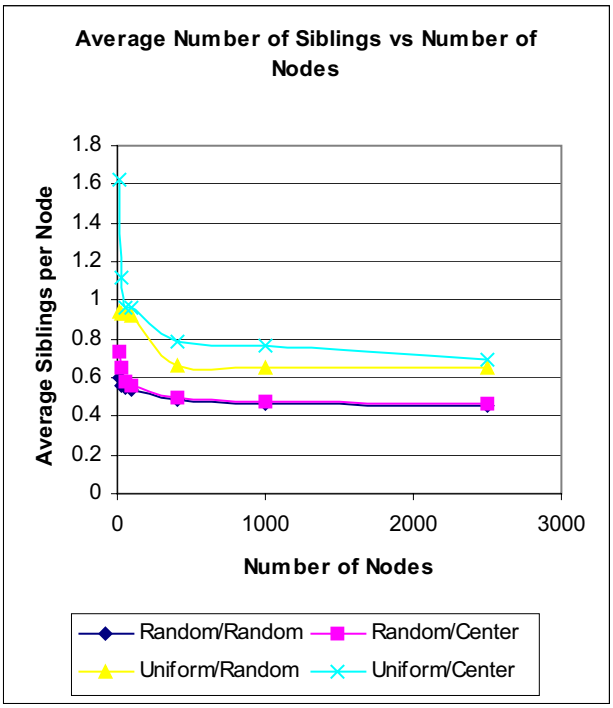

Figure 22. This graph compares average number of siblings per node as the number of nodes increases.

Having a cluster-head near the boundary is equivalent to a network with a higher number of nodes and the cluster-head at the center, when comparing average number of siblings. This can be seen geometrically by examining a randomly distributed network with density ρ. Let r be the average distance between nearest neighbours, and let A_h be the area of a ring with width r and hr from the cluster-head. Any nodes in A_{h+1} will, with high probability, select a node in the adjacent inner ring A_h. The estimated average number of siblings is for nodes in area A_h is:

$$\frac{\rho A_{h+1}}{\rho A_h} - 1 = \frac{\pi((h+1)r)^2 - \pi(hr)^2}{\pi(hr)^2 - \pi((h-1)r)^2} - 1$$

$$= \frac{2h+1}{2h-1} - 1$$

The average number of siblings for the entire network is:

$$\frac{\int_1^d \frac{2h+1}{2h-1} - 1 dh}{d} = \frac{d + \ln d - d}{d} = \frac{\ln d}{d}$$

As the height of the network, d, increases, the average number of siblings approach zero:

$$\lim_{d \to \infty} \frac{\ln d}{d} = 0$$

A.3. Proof of Theorem 3 – Time Complexity

The time complexity of HIT is the time to compute the TDMA schedule. Let us assume a network with depth D and average of S siblings per node. To simplify the analysis, a new uniform network is created, where every node j with height less than D has $\lceil S+1 \rceil$ downstream neighbors starting with a single cluster-head at height zero (Fig. 23). This new network establishes an upper bound on the time complexity of the scheduling algorithm for the original network. It follows from Lemma 1 that nodes on the same level can collide with each other if they share the same upstream neighbor but nodes having different upstream neighbors do not collide. Therefore the maximum number of concurrent transmission at some height h is equal to the number of nodes at height $h-1$. Since each node at level $h-1$ has S siblings, it takes only $S+1$ transmission times for all nodes at any level n to transmit their data.

Applying the schedule algorithm to the new network, it takes $S+1$ iterations of the outer while loop (1) to schedule each level in the network therefore

the total number of executions of the while loop $O((S+1)\times D) = O(S\times log(n))$. The inner loop (2) and (3) execute 2n times. So the total execution time is O(S×log(n)×2n) = O(S×n×log(n)). From Lemma 2 it follows that the average number of siblings decreases as n increases, $T=ceiling(S)+1 \approx 0+1 = 1$, and thus:

$$O(T\times n\times log(n)) = O(n\times log(n)).$$

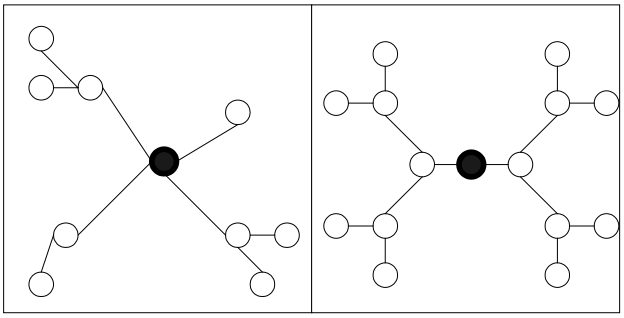

Figure 23. The figure on the left has an average of 1.8 siblings per node. The figure right is a uniform network where every node other than the cluster head has $\lceil 1.8 \rceil = 2$ siblings per node.