Dynamic Route Selection for Wireless Sensor Networks*

W. Ke and T.D.C. Little
Department of Electrical and Computer Engineering
Boston University, Boston, Massachusetts
{*ke,tdcl*}*@bu.edu*

May 1, 2006

MCL Technical Report No. 05-02-2006

Abstract–With the decrease in the cost and size of computing devices, wireless sensor networks (WSNETs) have the potential to be grow to enormous numbers of nodes. To be of utility to evolving missions, such systems must have the ability to be tasked, retasked, and multiply tasked. To support this vision, we investigate the coexistence of multiple routing schemes targeted towards diverse tasks with a goal towards energy efficiency in routing.

We posit that to fully tap into the potential of such networks a new routing infrastructure is needed that allows switching between different routing schemes dynamically as required by the applications being deployed, the conditions of the network as a whole, and the existing locality information. In this paper we shoe how dynamic routing scheme selection can be achieved when sensor networks are overlaid with a virtual attribute based cluster hierarchy. We present analytical results for our scheme and show the expected improvements achieved.

This work is supported by the NSF under grant No. CNS-0435353. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

Technological advances are enabling the deployment of wireless sensor networks for many different applications [1]. These applications are varied in scope and purpose, including object tracking [2], structural health monitoring [3], habitat monitoring [4] and monitoring of the environment and its resources [5]. In the future, with such sensors attached to cars, other electronic devices and human bodies, we will live immersed in a pervasive sensorsphere comprised of internetworked sensor network applications.

To utilize the resources existent in such sensorsphere, an appropriate routing infrastructure must exist. Routing paradigms that rely on flooding, such as Directed Diffusion [6], are expected to generate too much redundant traffic in the scenario above, since inquiries can be initiated by different users targeting subsets of deployed sensors. Geographic routing models such as GPSR [7] or GEAR [8] are essentially application independent. They require applications to discover or know *a priori* the location of specific data targets or a sink. Data-centric models built on top of these geographic models, such as GHT [9], DIM [10], DIFS [11] and DIMENSIONS [12] do not offer sufficient low-level control over communication patterns for supporting divergent applications and routing needs.

Given the large number of nodes in the future sensorsphere and the existence of redundant nodes and subsequent oversampling, there is some question about the utility of device-specific addressing commonly used in many routing protocols. If many sensors exist in an enclosure, a single representative value of a sensed modality can be sufficient to report, for example, temperature. Towards defining classes from which to draw these representative values, we propose establishing an attribute-based representation that is hierarchical; leveraging attributes of the sensors, location, and other identifiable class descriptions. The attributes chosen must satisfy locality-based relationships (we propose two: containment and adjacency relationships) and can reflect the frequency with which they are present in the inquiries. The clustering mechanism joins attribute equivalent sensors together. Clusterheads in this context act as attribute-based routers, and can support different routing schemes based on the application needs.

We show in this paper how routing schemes can be built on top of such attribute based hierarchical clustering system, and demonstrate, through theoretical analysis, how switching between different traversal modes of the clusters result in higher gains for different metrics. This difference in the gains for different metrics show that, for a sensorsphere as described above, a routing

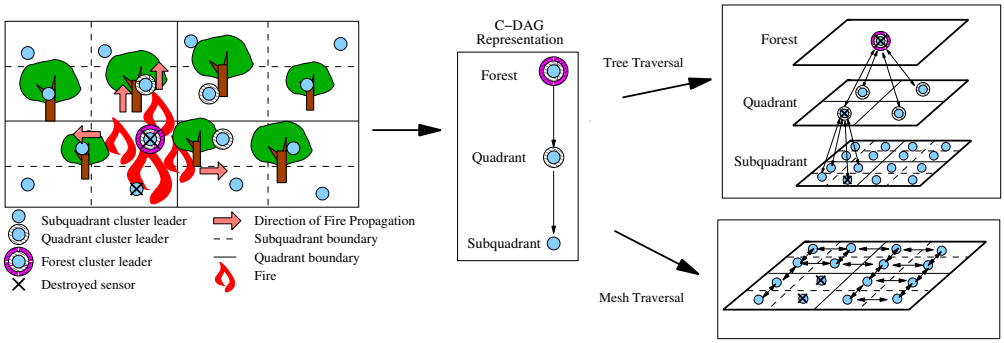

Figure 1: Sensors deployed in a forest under $\{Subquadrant \subset Quadrant \subset Forest\}$ attribute hierarchy and two modes of cluster traversal: *Tree* or *Mesh*.

infrastructure that can support dynamic selection of routing schemes is necessary for improved application-level performance.

The remainder of the paper is organized as follows. In Sec. 2 we review related work. Sec. 3 describes example scenarios of our scheme. We delineate our design choices and show the basic functionality specification, as well as data structure and algorithms related to the implementation of the routing infrastructure in Sec. 4. Theoretical performance analysis are presented in Sec. 5. Sec. 6 concludes the paper.

2 Related Work

Existing pproaches such as directed diffusion [6, 13] flood inquiries to the network and build gradients supporting reverse paths to sinks for data collection. Such approach is limiting for the general case of routing to support cases beyond a rooted tree or for a multitude of independent sensing tasks, particularly due to the flooding step.

To reduce the redundant transmissions of packets, location information is explored in order to direct how data can be routed. GPSR (Greedy Perimeter Stateless Routing [7]) and GEAR (Geographical and Energy Aware Routing [8]) are two examples of geographical-based routing. Both rely on the presence of a location service to operate, and in both the addressing scheme is independent of the applications they support. In other words, a data sink must know *a priori* the region to which send the data request and vice-versa. The existence and maintenance of a location service is non-trivial, especally in the presence of mobility, and any assumtion of its presence must be reviewed critically.

Semantic Routing Trees (SRT) are proposed in reference [14]. Tree structures are formed in the sensor network based on sensed values and queries are forwarded to children that have values within the range requested. Like SRT, but with a more generalized filtering approach, CBCB (Combined Broadcast and Content Based routing [15]) adopts a two layer approach (one broadcast layer and one content-based layer) to place predicates (a set of constraints on the attributes) at the routers. Data that match a predicate are be forwarded to appropriate sinks. Our work differs from SRT and CBCB in that we do not attempt filtering at sensor level, but instead form attribute equivalent regions that help route traffic. We also support different communication needs and patterns in such regions.

The advantages of being able to select the routing protocol at run-time have been established by the active network community [16]. Work in reference [17] proposes encapsulating packets in SAPF (Simple Active Packet Format) headers, which carry indicators to an active node's FIB (Forwarding Information Base), guiding packet forwarding behavior at run-time. The routing example shown in reference [17] is tree based. In reference [18] the authors propose an overlay scheme that allows active nodes to coexist with passive nodes. The active nodes track communication paths to each other reactively. Our work shows how dynamic routing protocol selection can be implemented in attribute clustered WSNETs. We show the routing rules and the performance analysis for both the tree and the mesh traversal modes. Furthermore, we show how the changing density of "active routers" (in our case attribute based routers or cluster leaders) in the network, achieved through changing the number of levels in the attribute hierarchy, affects the expected performance of the two routing schemes. Next, we will delineate some examples of scenarios that motivate our work.

3 Scenarios

Environmental Monitoring and Fire Annunciation Consider the following scenario: multimodal sensors are deployed over an area for climate monitoring, and are collecting average values of temperature and humidity when suddenly fire is detected. One local application, designed to detect and track how the fire propagates, is awakened and immediately alerts neighbor sensors so that the fire front can be detected. This scenario is depicted in Fig. 1.

The communication needs of the sensor network while in the first stage of monitoring average temperature and humidity can be thought of as hierarchical. Data are slowly aggregated within each cluster by the cluster leader and sent to the base station. Thus sensors communicate using the "Tree traversal" mode found on the upper right side of Fig. 1. However, the communication needs of the fire detection application add a new component: the necessity for clusters to communicate with neighbor clusters, so that the fire propagation can be tracked over time. The way the fire propagates is also recorded and this information is spread to contiguous clusters, as in the event of a fire there is no guarantee that the top hierarchical leader has survived the fire. This situation is also depicted in Fig. 1, in which the sensor which plays the role of Forest leader, as well as Quadrant SouthWest leader has been destroyed by the fire. If the tree traversal hierarchical mode is the only communication mode, then other quadrant leaders would not be able to detect the fire in time. However, by using the "Mesh traversal" mode (lower right side of Fig. 1) at the lowest level of the attribute hierarchy (Subquadrant clusters), sensors are able to spread the alarm and continue detecting the fire front.

The example above illustrates how different applications may require different communication patterns. It is possible and likely, given the sensors are multimodal [1], that other applications are also present, e.g., wildlife tracking (needs to be able to communicate with neighboring sensors, to alert them of the tracked object, and needs to be able to send logged data back to base station), which would further drive the need for a common, yet flexible routing infrastructure. The next scenario illustrates how supporting different addressing systems concurrently may be beneficial for application deployment.

Tracking and Management Sensors are used to identify, track, log movements, and raise breach of security alarms (a person being where it ought not to be) within a high security building. The tracking part of the application requires communication from sensor clusters with their neighbor clusters, while the information logging benefits from a hierarchical approach. The alarm feature benefits from a logical, organizational type of addressing system (e.g., "Multimedia Communications Lab") rather than a "pure" location-based addressing system ("445 Saint Mary's St.").

Suppose that within that building a temperature and lighting control and maintenance application is also being deployed on the same network. This application is not concerned with security issues and is only interested in regulating temperature and lighting conditions when a person is detected within a room. Temperature and lighting controls are local, so information sharing must be localized (room 445 may have no one while room 446 is hosting a party – it is useless for room 445 to know of the conditions in room 446 and vice-versa despite their being right next to each other) but in the event of a malfunction (temperature rises above 90 F when no one is in the room) the alert must be directed to the building manager's/maintenance office, which may not be in the same building. In this case, a hierarchical clustering scheme may facilitate locating the building manager/maintenance office. For the maintenance team, knowledge of the organizational address (e.g., "ECE Dept"') would not be as helpful as knowing the correct location ("8 Saint Mary's St.,

Rm 324”).

We can see from this example that in addition to supporting different routing schemes, an additional desirable feature of a routing infrastructure for sensor networks is its capacity to support multiple simultaneous addressing schemes. In the next section we show our design choices and a specification of required functionality.

4 Routing

In host(device)-centric routing schemes identifiers are given to network nodes that are independent of any attributes or data the host/device may possess. This approach may be justified when in a network the emphasis is in finding the device, that is, data sets of interest map to a relatively few hosts. However, in the situation in which multiple devices share common data sets, reaching a specific device is a less significant goal, and the reverse should be attempted: to reach the data of interest rather than a specific device (see Fig. 2).

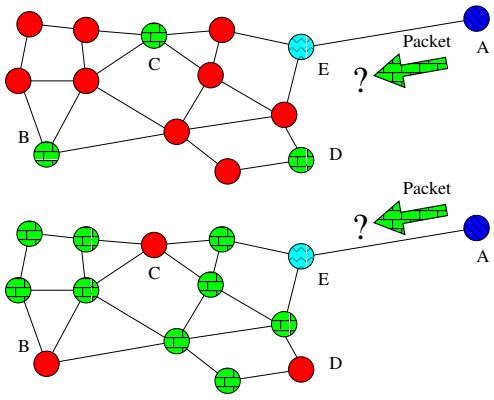

Figure 2: *Top*: data exists in a few hosts – packet sent by A needs to reach hosts B, C or D. *Bottom*: data replicated in almost all hosts – packet sent by A can retrieve data from any host except B, C, D or E.

The challenge in this situation is to propose a scheme that can identify data sets of interest at varying degrees of accuracy. We propose using an attribute hierarchy for this purpose. Attribute hierarchies can be determined by each sensor network locally, have flexible degrees of accuracy (ranging from including all sensors in a large geographic region to being able to pinpoint specific sensors, given enough attributes), can be easily manipulated to provide basic units on top of which routing takes place (e.g., routing between rooms, or floors, or buildings), and can be overlaid (two attribute hierarchies may be used by different applications to target the same set of sensors in different ways, e.g., routing may happen between rooms, floors and buildings, or offices, departments and colleges).

Next we summarize design considerations and characteristics of one mechanism that can provide a logical overlay of attribute based hierarchical clusters on the sensor network. More specific details can be found in [19].

4.1 Attribute Based Hierarchical Clustering

The set of attributes used to address sensors must satisfy containment relationships, that is, higher level attributes contain lower level attributes, and have all adjacency relationships defined, that is, which attribute values are spatially contiguous to each other. A hierarchy thus formed can be represented via a directed acyclic graph (DAG), in which nodes represent attributes and edges the containment relationships. Because of this we call these DAGs Containment DAGs or C-DAGs for short (see the C-DAG in the center of Fig. 1). Adjacency relationships are not formally represented by any specific data structure, but are exhaustively listed.

Nodes that have the same attributes are clustered together. This clustering occurs in a hierarchical manner. Each cluster represents an attribute-equivalent region, and by controlling which and how many attributes are part of the hierarchy we can control whether the propagation of inquiries is pure flooding (one level in the hierarchy), host centric (attributes have resolution that can pinpoint individual sensors uniquely), or a hybrid approach, in which we have attribute equivalent regions communicating with each other.

Once attribute-equivalent regions have been established, clusterheads can coordinate intra- and inter-cluster data dissemination based on the application requirements. Thus part of the network that is being tasked with an object tracking application may have different routing rules than another part which has been given the task of collecting soil humidity profile. Different performance expectations from the application may also result in different routing rules. We posit that attributes should be selected based on an *a priori* assumption on the frequency such attributes will be called by users in their inquiries. We support dynamic modifications to the C-DAG after deployment to insert or remove nodes to more efficiently guide data propagation. For more details on the clustering process, see [19].

4.2 Rules-Based Routing

In our proposed framework the routing process is an interpreted one. Nodes in a WSNET process incoming packets based on routing rules that are grouped by sets. Different sets of routing rules define different communication patterns (e.g., “tree” traversal, or “mesh” traversal). The initial set of routing rules is either present in the nodes before deployment or is propagated at cluster formation time. These are application independent sets. Nodes can hold different application independent sets simultaneously. The initial set is the default one. If an application needs to invoke other set of routing rules for packet processing, it must indicate so by adding a routing rule set identifier in the packet header. If the requested routing rule set is not present in the node, then default routing behavior is adopted.

Applications can bring with them their own set of routing rules, though, and these may be changed dynamically. If applications do not require change of routing rules at all nodes in the network, but only on a small subset, then they may request forming a small cluster for this purpose. Nodes that become members of such cluster either must possess the same routing rule set or request the set from the cluster leader. The purpose of these clusters is just to enable different communication patterns for a small subset of nodes, and thus no inherent support exists for managing high numbers of members. There is no limitation on the possible number of members, but a single cluster with many members will have significant performance degradation. By setting routing as an interpreted process, we allow dynamic configuration of nodes to support different communica-

Table 1: Different Routing Schemes – (a)*Left:* Tree traversal, (b)*Right:* Mesh traversal)

```
1:  CDAG ← {Subquadrant ⊂ Quadrant ⊂ Forest};
2:  RoutingTable ← Routing table used by current application;
3:  SensorAttributes ← Attributes current sensor possesses;
4:  SensorClusters ← Set of clusters the current sensor belongs to;
5:  SensorClusterLeader ← Set of clusters the current sensor is leader
    of;
6:  N(X, Y) = function that returns the number of consecutively
    matched attributes between X and Y, starting from the first at-
    tribute in both X and Y;
7:  Received packet P;
8:  DestAttrList ← list of attribute name-value pairs of the destination
    in P;
9:  Find E ∈ RoutingTable | (N(DestAttrList,E) is maximized) ;
10: if (E = DestAttrList) then
11:     if (E ∈ SensorClusters) then
12:         Flood P in E; Return;
13:     else if (P.PrevHop ∉ {path between current sensor ∧ E})
        then
14:         Send P to E; Return;
15:
16: if (∃ L ∈ SensorClusterLeader | (L = P.NextHop)) then
17:     if (P.PrevHop is parent node in CDAG) ∨ (sensor is root
        leader) then
18:         if (∃ children node | known attributes of children node match
            DestAttrList) then
19:             Send P to children node in CDAG;
20:         else
21:             Drop packet P;
22:     else
23:         if (∃ unmatched attribute at level L or higher between the
            sensor and DestAttrList) then
24:             Send P to parent of L;
25:         else if (all attributes from root to level L match between
            the sensor and DestAttrList ∧ ∃ child cluster with increased
            attribute match) then
26:             Send P to sibling clusters;
27:             Send P to child cluster;
28:         else
29:             Drop P;
30: else
31:     Send P to leader of P.NextHop;
```

```
1:  CDAG ← {Subquadrant ⊂ Quadrant ⊂ Forest};
2:  RoutingTable ← Routing table used by current application;
3:  SensorClusters ← Set of clusters the current sensor belongs to;
4:  SensorClusterLeader ← Set of clusters the current sensor is leader
    of;
5:  N(X, Y) = function that returns the number of consecutively
    matched attributes between X and Y, starting from the first at-
    tribute in both X and Y;
6:  Received packet P;
7:  if (P was received before) then
8:      Return;
9:  DestAttrList ← list of attribute name-value pairs of the destination
    in P;
10: Find E ∈ RoutingTable | (N(DestAttrList,E) is maximized) ;
11: if (E = DestAttrList) then
12:     if (E ∈ SensorClusters) then
13:         Flood P in E; Return;
14:     else if (P.PrevHop ∉ {path between current sensor ∧ E})
        then
15:         Send P to E; Return;
16:
17: if (∃ L ∈ SensorClusterLeader | (L = P.NextHop)) then
18:     if (∃ children node | known attributes of children node match
        DestAttrList) then
19:         Send P to children node in CDAG;
20:     else if (∃ adjacent cluster C at same level of L with matching
        attribute ∧ no copy of P came from C) then
21:         Forward P to all such C;
22:     else
23:         Drop P;
24: else
25:     Send P to leader of P.NextHop;
```

tion patterns and thus meet different communication needs from the various applications that share the network.

Each "rule" in our rules-based routing is comprised of two parts: (1) a conditional statement and (2) an action statement. If the conditions specified are true, then the action is performed. Otherwise, the following rule in the rule set is checked. If no conditional statement evaluates to true after all rules are exhausted, the packet is dropped. Our rules-based approach essentially imposes a priority scheme over possible next-hop destinations. Each conditional statement defines a subset of all possible incoming packet states, and each action statement essentially defines a possible next-hop destination. Thus the order in which the rules are placed within the rule set reflects the priority assigned to each possible state-destination association. Ideally, the first rule in the rule set should reflect the most common applicable rule in the network. Because of this "condition-action" separation, the rule set can actually be described by a series of **if-then-else** statements.

In the right side of Fig. 1 two communication patterns are established, the "Hierarchical Tree Traversal" mode, in which lower-level cluster leaders communicate with higher level cluster lead-

ers, routing in a hierarchical virtual tree, or the "Mesh Traversal" mode, in which cluster leaders at the lowest level in the C-DAG communicate with adjacent cluster leaders, routing on a logical mesh. Routing rules for both traversal modes are shown in Table 1.

The Mesh traversal algorithm (Table 1b), unlike the Tree traversal (Table 1a) one, drops packets that have been seen before (Line 8 in Table 1b). In the tree traversal, unknown destination packets may be sent to higher-level cluster leaders (Line 24 of Table 1a), and these may eventually forward the packets back (Line 27 of Table 1a). The Mesh traversal algorithm forwards packets of unresolved attributes to neighbor clusters (Line 21 in Table 1b). Notice the different approach each routing rule establishes on resolving unknown addresses: while in the tree case the packets are forwarded up the hierarchy, in the mesh the packets are simply spread towards other adjacent clusters. These two resolution modes also characterize the intrinsic communication pattern each rules set supports. Sensor networks that are deployed for different applications will benefit from being able to support switching between the two modes, as we show in the next section.

5 Performance Analysis

In this section we study the performance of the two routing schemes represented by Table 1 as applied to a "line" C-DAG (center of Fig. 1), and of schemes that rely on "flooding" for data propagation, as well as schemes that have full knowledge of all sensors in the network. We divide the Tree traversal scheme into two: one in which cluster leaders track only information of their children cluster leaders (labeled "Tree (One level information)"), while in the other scheme the cluster leader tracks information of all its cluster members (labeled "Tree (Full cluster information)").

The network consists of N sensors spread uniformly over a square region of area L^2 and there are l_h levels in the line attribute hierarchy. Since the C-DAG representation of the attribute hierarchy is a line, there are l_h nodes in the C-DAG. The root node (at level 1) in the C-DAG covers the whole region, while subsequent nodes (at levels l_i, $i \in \{2, \ldots, l_h\}$) have four possible values each (a quadtree format), with each value covering a square region of side $L/2^{(i-1)}$. In the right of Fig. 1 a three level C-DAG is shown.

The metrics studied for each scheme include: (1) total memory requirement from all nodes for implementation; (2) the estimated number of transmissions taken when routing one packet from a source to an unknown destination in the worst case (considering that the sensors are deployed over a square region, the worst case is when source and destination lie at opposite corners across a diagonal) and (3) the estimated number of hops that separate source from destination after the destination's address has been "resolved" in (2). Essentially (1) allows us to gauge how scalable each scheme is in terms of the amount of memory needed. Metric (2) allows us to compare the cost of resolving an unknown destination address, while (3) is an estimate of how quickly the destination address can be found or how quickly data can be transmitted to the destination, assuming both being directly proportional to the hop distance that separates source from destination.

When estimating item (2) and (3) above, for non-flooding type of schemes, we consider that the path the packet takes is composed of consecutive straight line segments. One estimate of the number of transmissions (or the number of hops) is the product of the length of the segment by the linear node density. The node density is given by $\rho = N/L^2$, thus one estimate of the number of neighbors that lie on a line segment within transmission radius R is $R\sqrt{\rho}$. On the average, assuming the sensors are uniformly distributed and the whole network connected, the number of

transmissions should not be greater than this value, for this value reflects the number nodes that lie in the segment. If this value is $\gg 1$, then we are overestimating the number of transmissions needed. Estimates made in this way can still be used for comparison between different routing schemes, though, since the overestimation comes from the high node density value and will be reflected by all routing schemes.

An estimate that is closer to the minimum number of transmissions (or number of hops) needed to cover the path between source and destination is obtained by dividing the path length by the transmission range R. However, when the "line" C-DAG has a very high number of nodes (i.e., high l_h), the leaf node's covered region may be smaller than the transmission range ($L/2^{(i-1)} \ll R$, when $i \gg 1$). Because our hierarchical routing scheme stores routing information based on attribute regions, and routes according to containment and adjacency relationships, the lower bound in the number of transmissions is the number of attribute regions traversed. The results of our performance comparison are summarized in Table 2.

Table 2: Performance Metrics for different Routing Schemes

	Flooding	Full	Tree (One level information)	Tree (Full cluster information)	Mesh
Memory	0	$E\,N(N-1)$	$E\,\frac{4}{3}\,(4^{(l_h-1)}-1)+E\,N\,l_h$	$E\,2\,N\,l_h$	$E\,(4^{l_h}+N+2(2^{(l_h-1)}-1)\sqrt{N})$
Max Num Tx	N	$\sqrt{2N}$	$\sqrt{N}\,(2^{(l_h-1)}-1)(\frac{2\sqrt{2}}{2^{(l_h-1)}}+\frac{3\sqrt{2}}{2}+\sqrt{5})+\frac{N}{2^{(2l_h-2)}}$	$4\sqrt{2N}(1-\frac{1}{2^{(l_h-1)}})+\frac{N}{2^{(2l_h-2)}}$	$2\sqrt{2N}(2^{l_h}-\frac{2}{2^{(l_h-1)}})+\sqrt{N}(\frac{8-4\sqrt{2}}{2^{(l_h-1)}})+\frac{N}{2^{(2l_h-2)}}$
Min	N	$L\,\sqrt{2}/R$	$\max(\frac{L}{R}\,(2^{(l_h-1)}-1)(\frac{2\sqrt{2}}{2^{(l_h-1)}}+\frac{3\sqrt{2}}{2}+\sqrt{5}),\sum_{i=2}^{l_h}(4^{(i-2)}\lceil\frac{L\sqrt{2}}{R2^{(i-1)}}\rceil+4^{(i-2)}\,2\lceil\frac{L\sqrt{5}}{R2^{(i-1)}}\rceil+(4^{(i-2)}+1)\lceil\frac{L\sqrt{2}}{R2^{(i-2)}}\rceil))+\frac{N}{2^{(2l_h-2)}}$	$\max(\frac{4L\sqrt{2}}{R}(1-\frac{1}{2^{l_h-1}}),\sum_{i=2}^{l_h}2\lceil\frac{L\sqrt{2}}{R2^{(i-2)}}\rceil)+\frac{N}{2^{(2l_h-2)}}$	$\max(\frac{L2\sqrt{2}}{R}(2^{l_h}-\frac{2}{2^{(l_h-1)}})+\frac{L(8-4\sqrt{2})}{R\,2^{(l_h-1)}},2^{l_h}\,(2^{(l_h-1)}-1))+\frac{N}{2^{(2l_h-2)}}$
Max Num Hops	$\sqrt{2N}$	$\sqrt{2N}$	$4\sqrt{2N}(1-\frac{1}{2^{(l_h-1)}})$		$2\sqrt{2N}(1-\frac{1}{2^{(l_h-1)}})+\frac{\sqrt{N}(4-2\sqrt{2})}{2^{(l_h-1)}}$
Min	$L\,\sqrt{2}/R$	$L\,\sqrt{2}/R$	$\max(\frac{4L\sqrt{2}}{R}(1-\frac{1}{2^{l_h-1}}),\sum_{i=2}^{l_h}2\lceil\frac{L\sqrt{2}}{R2^{(i-2)}}\rceil)$		$\max(\frac{L}{R}(2\sqrt{2}(1-\frac{1}{2^{(l_h-1)}})+\frac{4-2\sqrt{2}}{2^{(l_h-1)}}),2\,(2^{(l_h-1)}-1))$

Flooding A flooding-based routing scheme is not required to store routing information about the network. Each packet is flooded to the entire network. Consequently, the memory requirement is zero[1] and it takes N transmissions to deliver the packet. The farthest any two sensors may be from each other is if they lie at opposite corners across a diagonal. Thus, transmission across the diagonal must cross a minimum of $L\sqrt{2}/R$ hops. Given the node density of $\rho = N/L^2$, then an estimate of the number of nodes lying in the diagonal is $L\sqrt{2\rho} = \sqrt{2N}$, and this is, on the average, the maximum number of hops that separates source and destination.

Full Knowledge A routing scheme that stores next-hop routing information for all nodes in the network has a huge memory requirement. In fact, each node needs to store information about $N-1$ other nodes in the network. Considering that each routing entry requires E bytes, the total

[1]Diffusion schemes are not purely flooding schemes. Interest propagation (setup) uses flooding followed by a hop-by-hop reverse path routing scheme to a sink.

memory requirement in the network is $E\,N(N-1)$. However, because of the complete knowledge, the number of transmissions triggered and the number of transmissions required to send the packet are equal. These are equal to the estimated maximum and minimum number of hops in the flooding case.

Cluster Flooding In both *Flooding* and *Full Knowledge* schemes destination sensors are sure to be reached. In "Tree" or "Mesh" schemes below, however, packets reaching the intended leaf cluster(s) still need to reach the sensors. Assuming the intended destination address "resolves" into one leaf cluster, to flood that cluster the number of additional transmissions is equal to $\rho\,(L/2^{(l_h-1)})^2 = N/2^{(2l_h-2)}$ is needed. This term appears in all "NumTx" entries in Table 2.

Tree (one level information) In our clustered hierarchical scheme, each node that is not a cluster leader tracks leaders of the cluster it belongs across all hierarchy levels ($E\,N\,l_h$). Assuming one cluster per attribute value, we have one cluster for the root node, four clusters for the node at the second level, 16 for the node at the third level, etc. Each cluster leader tracks the routing information of its four children clusters. Leaf cluster leaders track information about their cluster members. Since leaf clusters cover the whole network, it requires N entries. Thus it is $E4(1 + 4 + 4^2 + ... + 4^{l_h-2}) + EN$ = $E4(4^{l_h-1} - 1)/3 + EN$. The sum of the two factors shown in this paragraph is the memory requirement equation for "Tree (one level information)" in Table 2.

When a packet with an unknown destination is received, it is sent to the cluster leaders through the hierarchy all the way up to the root node if no matching attributes are found. The longest segment that separates the root to a second level cluster leader is $L\sqrt{2}$, while the longest segment that separates the second level cluster leader to a third level child cluster is $L\sqrt{2}/2$. Thus the sum of the segment lengths is at most U_L = $L\sqrt{2}(1 + 1/2 + \cdots + 1/2^{(l_h-2)})$ = $L2\sqrt{2}(1 - 1/2^{(l_h-1)})$

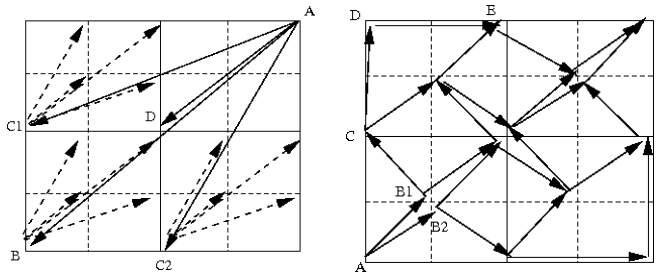

Figure 3: (a) *Left:* Propagation path for *Tree* traversal when resolving unknown destination address (b) *Right:* Propagation path for *Mesh* mode when resolving unknown destination address

Also, assuming the packet reaches the root node, the root node must send the packet downwards to all of its child clusters, which each may in turn pass it down again, eventually reaching the leaf cluster leaders, at which point the cluster leader that satisfies the destination address attributes floods the packet to its cluster. In this process of forwarding the packet down the C-DAG, from the root node to the second-level cluster leaders four segments are required (covering the longest distance possible): the longest will be $L\sqrt{2}$ (segment AB in Fig. 3(a)), while there will be two segments of $L\sqrt{5}/2$ (segments $AC1$ and $AC2$ in Fig. 3(a)), and one segment of $L\sqrt{2}/2$ (segment AD in Fig. 3(a)). From the second-level cluster leader to the third-level clusters the same process will be repeated: the packet is sent to four clusters, with the longest segment being half of the longest segment of the previous level, two segments which are half of the two analogous segments of the previous level, and the shortest segment being half of the shortest in the previous level

(shown as dotted lines starting from $B, C1, C2$ - omitted for D for clarity's sake). This process repeats itself all the way down to the clusters at level $l_h - 1$.

Thus, assuming $S = (L\sqrt{2} + 2L\sqrt{5}/2 + L\sqrt{2}/2)$, then the total segment length the packet may need to traverse when going "down" is $D_L = S + 4S/2 + 16S/4 + \cdots + 4^{(l_h-2)}S/2^{(l_h-2)} = S + 2S + \cdots + 2^{(l_h-2)}S = S(2^{(l_h-1)} - 1) = L(2^{(l_h-1)} - 1)(3\sqrt{2}/2 + \sqrt{5})$. The total length is then $T_L = U_L + D_L = L(2^{(l_h-1)} - 1)(2\sqrt{2}/2^{(l_h-1)} + 3\sqrt{2}/2 + \sqrt{5}$.

Of the four packets that are sent from a higher level leader to a lower level leader only one eventually reaches the destination. So that we will estimate the number of hop that can cover the worst case (i.e., sender and receiver are farthest apart), we take the longest segment at each level, and thus $T_L = 2U_L = L4\sqrt{2}(1 - 1/2^{(l_h-1)})$.

The higher estimate on the maximum number of transmissions is obtained by multiplying the length obtained by $\sqrt{N}/L$ (see "NumTxMax" and "NumHopsMax" equations in Table 2), while an estimate on the minimum number of transmissions is obtained by dividing the length by R (see the "NumTxMin" and "NumHopsMin" equations for $L/(R2^{(l_h-1)}) \gg 1$ in Table 2).

However, in the case in which the transmission range is much higher than the leaf attribute region side, the number of transmissions is lower bounded by the number of attribute regions the packet crosses. Given that there are four different segments that the root node needs to send to reach the level 2 leaders, each of the two segments of equal length ($AC1$ and $AC2$ in Fig. 3(a)) will generate $4^{(i-2)}$ segments of length $L\sqrt{5}/(R2^{(i-1)})$ at level $i \geq 2$. In the same way we count $4^{(i-2)}$ segments for the shortest segment (AD in Fig. 3(a)) and $4^{(i-2)} + 1$ for the longest. The "+1" is because we must also count the transmission costs incurred when the packet was coming up the hierarchy towards the root node. Each segment counts at least once (i.e., we round up the cost) no matter how small its length is with respect to the transmission radius R, because it represents one distinct attribute region. When we sum up for all levels in the hierarchy we obtain the corresponding expression in Table 2.

When we consider the number of hops that can separate source from destination, the worst case occurs when the source and destination are resolved by traversing the longest segment across all levels of the hierarchy. This is represented by the corresponding equation in Table 2.

Tree (full cluster information) For a tree scheme in which cluster leaders track all information from its cluster members the following memory requirement is necessary for a cluster leader at level i: $\rho\,(L/2^{(i-1)})^2 = N/2^{(2i-2)}$. Since this is a quadtree format, there are exactly $4^{(i-1)}$ children cluster leaders at level i, thus the memory requirement for cluster leaders to track cluster member information is exactly $l_h N$. Adding this to the requirement of N nodes tracking their l_h cluster leaders, the total memory requirement is $2l_h N$, as seen in Table 2. For the tree traversal mode with full cluster information, it is not necessary for the root node to forward the packet down to all of its children clusters. Since it has information of all the sensors in the network, it can forward the packet to the child cluster that contains the desired destination attributes. Thus the number of transmissions and the number of hops in this case is the same, and it corresponds to the case in which the longest segment is taken both when the packet is coming up the hierarchy and going down the hierarchy to the destination leaf cluster.

Mesh We study the performance of a routing scheme in a mesh like topology at only one attribute hierarchy level (say l_h). In a mesh like routing scheme, we assume each cluster leader tracks only its (at most) four neighbor clusters, resulting in a memory requirement of $E\,4\,4^{(l_h-1)}$. Also all sensors track their cluster leader ($E\,N$ of memory), and sensors that lie at the attribute border will track the two clusters for which it is the border. At level l_h, the total length of the border is

$2(2^{(l_h-1)}-1)L$, which, when multiplied by $\sqrt{N}/L$ and summed with the other terms, results in the memory requirement equation seen in Table 2.

We assume that when a packet with an unknown destination is received it will be transmitted to the neighbor clusters other than the ones from which the packet arrived. Thus if a packet is sent from the lower left cluster leader, with a destination that is unknown to the cluster leader, but whose final sink is in the top right cluster, then the packet will be propagated across all attribute regions. The total length traversed as the packet is distributed in the network is longer if the cluster leaders are located close to opposite corners across the diagonal, in the zigzag pattern shown in Fig. 3(b). In this figure we show the traversal taken when there are three nodes in the line C-DAG. The cluster leader of the lower left attribute region (A) sends the packet to its immediate neighbor cluster leaders ($B1$ and $B2$). As these are located close to the corner across the diagonal the length traversed is $2L\sqrt{2}/2^{(l_h-1)}$. To increase the length traversed, as the packet gets closer to the top left and bottom right corners, we assume the cluster leaders are located at the corners of their respective attribute regions. In this way we force comparison of the worst case in a mesh approach with the worst case of the tree based scheme analyzed previously. Notice that essentially the packet traverse the diagonals of squares with side length $2jL/2^{(i-1)}, j \in \{1, 2, 3, ..., 2^{(i-1)}\}$ in a regular fashion, discounting the borders and the top left and bottom right corners. The total length traversed, and the corresponding expected number of transmissions (both maximum and minimum) are given by the corresponding expressions in Table 4.

When the transmission radius $R \gg L/2^{(i-1)}$, then it takes at least one transmission to cross one attribute region, and assuming each attribute region will transmit to two of its immediate neighbors (with top and right border attribute regions transmitting only once), the total number of transmissions will be $2(2^{(l_h-1)}-1) + 2(2^{(l_h-1)}-1)^2 = 2^{l_h}(2^{(l_h-1)}-1)$, as seen in the table.

The shortest path that separates the source from the destination must traverse $2(2^{(l_h-1)}-1)+1$ attribute regions (the $+1$ is because the source attribute region also must be traversed). However, if the packet goes through only the diagonals, only $2(2^{(l_h-1)}-1)$ diagonals need be crossed. One of the attribute region leaders will receive the packet from the left and can immediately forward to the upper region, without needing to traverse itself. Thus the worst case scenario is actually when the source is at the top left corner while the destination is at the bottom right corner (or vice-versa). In this case there are additional four traversals across the border of the attribute region ($4(L/2^{(l_h-1)})$) and two less diagonal traversals. This explains the second term in the NumHopMax and the second term in the first argument to the $\max$ function in NumHopMin. When we are considering the minimum number of hops, this must be lower bounded by the number of attribute regions that need be crossed ($2(2^{(l_h-1)}-1)$), since in principle the cluster leader only tracks the four adjacent clusters. We show some plots of the equations of Table 2 in Fig. 4.

We see from Fig. 4(a) and Fig. 4(b) that the expected number of transmissions to resolve an unknown address in the worst case is higher for the mesh traversal mode than for the tree cases. When cluster leaders track full cluster information, the performance dramatically improves. This is because the root node need not propagate the packet with unknown address down to its children clusters. We see that the high number of levels in the attribute hierarchy contributes to the inefficiency of the process (Fig. 4(b) and 4(e)). With the increase in the number of hierarchies, the packet with unknown destination address need essentially be distributed to the whole network in the mesh and tree (with one level information) schemes at increasing levels of granularity (i.e., covering more of the network), contributing to their performance degradation.

A high number of levels will involve transmission costs to cross adjacent clusters in the mesh

case and costs to resolve all the way to the leaf cluster in the Tree (one level info) case. These costs surpass those of the mere flooding schemes and should be avoided. The cost for resolving an unknown address in the tree (full cluster info) case remains constant. However, the memory requirements are high (Figs. 4(c) and 4(f)).

When we consider the number of hops metric, we find that mesh schemes are able to find shorter paths between source and destination. The only drawback is that mesh schemes currently only cross spatially adjacent attribute regions. Thus when the number of levels in the hierarchy increases, there is a corresponding increase in the hop distance (Figs. 4(h) and 4(i)).

From the graphs in Fig. 4 we see that if the network is composed of heterogeneous nodes, in which some nodes have higher capacity, then a Tree (full cluster info) scheme will be the most economical in transmission costs related to address resolution issues. Sensor networks that have a high inquiry arrival, especially from a large user base, will benefit from the increased savings in Tree based address resolution schemes, while applications that require fast response can invoke mesh traversal mode for their data packets.

6 Conclusion

In this paper we present examples in which multiple applications are tasked to a single sensor network. Wth differing objectives, the underlying data communication and dissemination patterns of these applications favor different routing schemes. We envision that sensor networks will be widespread in the future. and to tap into the full potential of such sensorsphere, the underlying routing infrastructure must support dynamic routing scheme selection for these applications. With a dynamic capability, tasked applications can tailor their behavior a routing scheme and improve their performance.

In order to enable dynamic routing scheme selection, we propose the application of sets of predefined routing rules that forward data as selected at runtime. The model assumes that the underlying sensor network has been clustered according to a hierarchy of attributes, and that containment and adjacency relationships between the clusters (or the attributes) are clearly defined. We present two routing rule sets for applications deployed in a sensor network with the above mentioned logical structure. One rule set implements tree traversal mode while the other mesh traversal mode. We show analytical performance results of the two traversal modes and show that the mesh traversal mode favors applications that need fast response, while tree traversal mode has less transmission cost when resolving a previously unknown destination.

References

[1] J. Hill and D.E. Culler. MICA: A Wireless Platform For Deeply Embedded Networks. *IEEE Micro*, 22(6):12–24, Nov/Dec 2002.

[2] R. R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed Target Classification and Tracking in Sensor Networks. *Proc. of the IEEE*, 91(8), August 2003.

[3] Jeongyeup Paek, Nupur Kothari, Krishna Chintalapudi, Sumit Rangwala, Ning Xu, John Caffrey, Ramesh Govindan, Sami Masri, John Wallace, and Daniel Whang. The Performance

of a Wireless Sensor Network for Structural Health Monitoring. In *2nd European Workshop on Wireless Sensor Networks*, Istanbul, Turkey, Jan 31 – Feb 2 2005.

[4] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. 1st ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 88–97, 2002.

[5] Rachel Cardell-Oliver, Keith Smettem, Mark Kranz, and Kevin Mayer. Field Testing a Wireless Sensor Network for Reactive Environmental Monitoring. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing ISSNIP-04*, December 2004.

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proc. 5th ACM MobiCom Conference*, Seattle, WA, August 1999.

[7] B. Karp and H. T. Kung. Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th ACM MobiCom Conference*, pages 243–254, Boston, MA, August 2000.

[8] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks. Technical report, UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001.

[9] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage. In *Proc. 1st ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, September 2002.

[10] Xin Li, Young Jin Kim, Ramesh Govindan, and Wei Hong. Multi-dimensional Range Queries in Sensor Networks. In *Proc. 1st ACM Intl. Conference on Embedded Networked Sensor Systems (Sensys'03)*, Los Angeles, CA, USA, November 2003.

[11] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy, and S. Shenker. DIFS: A Distributed Index for Features in Sensor Networks. In *Proc. 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications (SNPA'03)*, 2003.

[12] D. Ganesan, D. Estrin, and J. Heidemann. DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks? In *Proc. 1st Workshop on Hot Topics In Networks (HotNets-I)*, Princeton, NJ, October 2002.

[13] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proc. International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, August 2000.

[14] S. Madden, M Franklin, J. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *Proc. ACM SIGMOD*, San Diego, CA, June. 2003.

[15] Antonio Carzaniga, Matthew J. Rutherford, and Alexander L. Wolf. A Routing Scheme for Content-Based Networking. In *Proc. IEEE INFOCOM'04*, Hong Kong, China, 2004.

[16] Christian Prehofer and Qing Wei. Active Networks for 4G Mobile Communication: Motivation, Architecture, and Application Scenarios. In *Proc. IFIP-TC6 International Working Conference (IWAN'02)*, London, UK, 2002.

[17] Christian Tschudin, Henrik Gulbrandsen, and Henrik Lundgren. Active routing for ad-hoc networks. *IEEE Communications Magazine, Special issue on Active and Programmable Networks*, April 2000.

[18] S. Calomme and G. Leduc. Performance Study of an Overlay Approach to Active Routing in Ad Hoc Networks. In *Proc. 3rd Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2004)*, Bodrum, Turkey, Jun 2004.

[19] Ke Wang, Salma Abu Ayyash, Prithwish Basu, and Thomas D. C. Little. Attribute-Based Clustering for Information Dissemination in Wireless Sensor Networks. In *Proc. 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05)*, Sant Clara, CA, USA, September 2005.

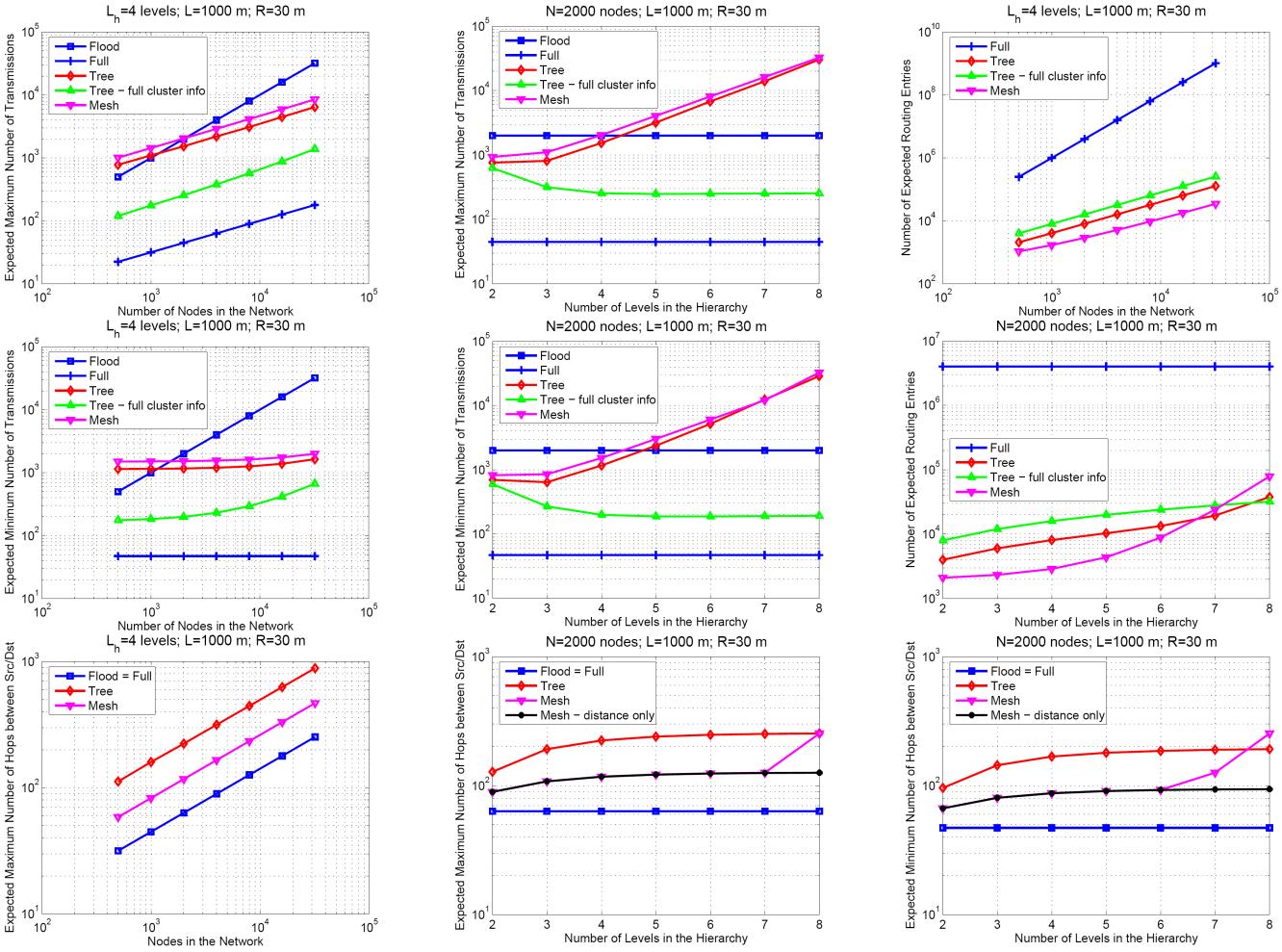

Figure 4: (a) *Top,Left:* Graph of expected maximum number of transmissions ($NumTxMax$) triggered by a packet with unknown destination (b) *Top,Center:* Graph of how $NumTxMax$ varies according to the number of levels in the hierarchy (c) *Top,Right:* Graph of Memory requirements with increasing number of nodes in the network (d) *Center,Left:* Graph of expected minimum number of transmissions ($NumTxMin$) triggered by a packet with unknown destination (e) *Center,Center:* Graph of how $NumTxMin$ varies according to the number of levels in the hierarchy (f) *Center,Right:* Graph of how memory requirements varies according to the number of levels in the hierarchy (g) *Bottom,Left:* Graph of expected maximum number of hops ($NumHopMax$) that separates source from destination as the number of nodes in the network increases (h) *Bottom,Center:* Graph of how $NumHopMax$ varies according to the number levels in the hierarchy (i) *Bottom,Right:* Graph of expected minimum number of hops ($NumHopMin$) that separates source from destination as the number of levels in the hierarchy changes