Field Division Routing

Milenko Drinić
Microsoft Corporation
mdrinic@microsoft.com

Darko Kirovski
Microsoft Research
darkok@microsoft.com

Gang Qu
and Lin Yuan
University of Maryland
{gangqu,yuanl}@eng.umd.edu

Miodrag Potkonjak
Computer Science Dept
UCLA
miodrag@cs.ucla.edu

***Abstract*—Multi-hop communication objectives and constraints impose a set of challenging requirements that create difficult conditions for simultaneous optimization of features such as scalability and performance. We have developed field division routing (FDR), a distributed and non-hierarchical routing protocol that aims to coordinated addressing of scalability, topology alternations, latency, throughput, energy efficiency, and local storage requirements. FDR is based upon two optimization mechanisms: a reactive and focused diffusion that collects only network topology information directly required for making localized routing decisions, and a protocol for sharing routing information among neighboring nodes. Routing table initialization and maintenance are scalable in terms of both storage and overhead traffic. FDR provides guaranteed connectivity while providing near-optimal all-node-pairs message delivery. The protocol is also power-efficient to a wide spectrum of topology changes that induce relatively few messages to update routing tables network-wide. We analyzed FDR both theoretically and using simulation.**

***Index Terms*—Ad-hoc networks, routing, multi-hop communication.**

I. Introduction

Wireless ad-hoc networks (WAHNs) are commonly abstracted as a system of application-specific devices (nodes) with processing, storage, communication, and often, sensing capabilities where nodes communicate via radio. Each node acts as a router and communicates with other nodes via a multi-hop protocol. Routing protocols in WAHNs play an important role as they have ramifications on the key system requirements: (*i*) scalability, (*ii*) resilience to topology change, (*iii*) overall node connectivity, communication (*iv*) latency (delay) and (*v*) throughput, (*vi*) power consumption, and (*vii*) local storage requirements [1]. Routing decisions in WAHNs are supported either using routing tables that specify the network topology or using geographic routing criteria for making localized forwarding decisions. The first class of protocols requires hard-to-scale worldwide routing table updates upon each topology change, while guaranteed message delivery and shortest-path communication are the key challenges for the latter class.

A popular heuristic approach that addresses criteria (*iii-vi*) is to ensure shortest-path routing between any two nodes in the network that can potentially communicate while minimizing the storage requirement at each node. All-shortest-paths routing is difficult to achieve using a scalable distributed protocol in the presence of arbitrary topology changes. We aim for this objective and propose a novel, distributed protocol, *field division routing* (FDR), that targets all of the seven system requirements. In FDR, each node contains all information necessary to distribute messages to any other node in the network; this information is derived based upon the geographic location of a relatively small but strategically located subset of nodes. Each node divides the geographical network field into a tile of zones unique for this node. Traffic to all nodes in one zone is routed via a neighbor uniquely assigned to that zone. If all FDR tables are computed centrally, they enable all-shortest-paths routing in a network with static topology [2]. Here, we explore the more complex but significantly more energy-efficient and scalable approach where each node computes and maintains its own routing table in the presence of an ever-changing network topology.

We introduce a DISCOVERY protocol that computes a near-optimal shortest-paths routing table for a given node. Since DISCOVERY must be repeated for each node upon every topology change, we reduce the network maintenance overhead by introducing a novel and near-optimal procedure for routing table INHERITANCE from neighboring nodes. Next, we propose a distributed protocol for network initialization with an objective to reduce the number of nodes that perform DISCOVERY. In order to support a dynamic network topology, we introduce algorithms for routing table updates that address node appearance and disappearance. We believe that FDR addresses the key system requirements (*i-vii*) simultaneously; it is scalable, establishes provable connectivity, is power-efficient because network dynamics initiates few messages to update routing tables, enables near-optimal shortest-path message delivery, and its routing tables are compact.

II. Related Routing Protocols

Routing protocols for WAHNs can be categorized as proactive, reactive, or hybrid. Each node that uses a proactive protocol, stores sufficient data about the network topology in its routing table. Routing tables are periodically updated such that any time a node needs to send a packet, the forwarding route is already known and can be immediately used [3]. In reactive protocols, a node initiates route discovery only when it forwards a packet. Frequently used routes are cached. Reactive protocols are suitable for highly dynamic networks where node mobility renders the cost of proactive protocols prohibitive [4]. The topology of WAHNs is closely related to the relative positions of nodes. Geographically assisted protocols exploit this property by making localized decisions on forwarding routes. Greedy Perimeter Stateless Routing (GPSR) [5] reduces the network topology to a planar graph. An extension to GPSR deals with realistic connections that often do not correspond to typically assumed unit graph network representations [6]. Hybrid protocols leverage on the advantages of both pro- and reactive routing schemes. Zonal protocols use proactive routing within a zone of limited scope and reactive routing on a global level [7]. DDR divides a network into non-overlapping zones and uses zone identifiers to effectively forward messages between zones [8].

FDR is a proactive hybrid non-hierarchical ("flat") protocol. At each node, FDR stores and updates only information that is necessary to make routing decisions at that node without storing the entire network topology. Simultaneously, it provides each node with the ability to forward messages to any part of the network. In addition, FDR is resilient to irregularities of nodes' communication radius [6], [9]. Since FDR uses network topology to derive routing tables, these irregularities affect only the shape of the routing zones.

III. FDR – Preliminaries

We constrain geographically the considered class of networks to a limited area of arbitrary shape called network field. A set of N nodes $\mathcal{N} = \{n_1, \ldots, n_N\}$ is distributed within the network field, and each node is aware of its location via location discovery algorithm [10]. Two nodes (*neighbors*) can communicate only if the Euclidean distance between them is smaller than their communication range and there are no obstacles to their communication. For brevity, we adopt that the communication range of all nodes in the network is a constant, r. From the viewpoint of a single node n_i, the network field is tiled into *routing zones* $\mathcal{Z}^i = \{z^i_1, \ldots, z^i_{Z_i}\}$. Each set of routing zones is node-specific. Each zone in $\mathcal{Z}^i$ is a polygon of arbitrary shape with one corner positioned at n_i. Each zone in $\mathcal{Z}^i$ is assigned exactly one neighbor to n_i.

Definition 1: **Routing Neighbor.** Exactly one neighbor to n_i is assigned to each of its routing zones. We define such a neighbor as a routing neighbor.

Each routing zone is defined with up to two *borders*. Each border is defined using a set of straight chain-connected *border line segments*

(abbr. line), connecting a pair of *border points*. A zone is defined using two borders that originate at n_i and intersect as a final point the border of the network field. The field border connects borders' ends to form a single polygon. In a less frequent case (0.0005% of times in our experiments), a routing zone does not extend to the border of the network field. In such case, a routing zone is described using a single border which starts and ends at n_i.

An example of a WAHN is shown in Figure 1. We consider the routing zones built from the viewpoint of node s. Node s divides the network field into three routing zones. These zones are assigned to neighboring nodes a, b, and c. The neighboring node a is assigned to the routing zone described with borders $\{(s, x_1), (x_1, x_2)\}$ and $\{(s, x_3)\}$; node b is assigned to the zone described with $\{(s, x_3)\}$ and $\{(s, x_4)\}$; node c is assigned to the zone described with $\{(s, x_4)\}$ and $\{(s, x_1), (x_1, x_2)\}$. Full description of each routing zone includes the border of the network. We assume that each node knows the defining points of the enclosing network field: y_1, y_2, y_3, and y_4.

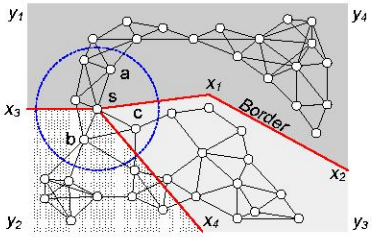

Fig. 1. An example of a network field divided into three routing zones for node s. Each zone is assigned to respective neighbor nodes a, b, and c of s. Routing zones are defined using points x_1, x_2, x_3, and x_4.

IV. Routing Table Initialization

During initialization, each node divides the network field into routing zones and assigns a routing neighbor to each routing zone. The construction process relies on several key observations related to network topology and connectivity.

Definition 2: **Essential Node.** If the shortest path from a source node n_i to a destination node n_j leads exclusively via one neighbor n_k of n_i, then n_j is a node essential to n_i.

Definition 3: **Essential Neighbor.** Let n_j be an essential node to n_i. A neighbor n_k of n_i, which is the first hop on the shortest path from n_i to n_j, is referred to as an essential neighbor of n_i.

Consider the example in Figure 2(a). Node s is the source. Then, nodes a_2 and a_3 are essential nodes because the shortest path to these nodes leads only via node a_1. Similarly, b_2, b_3, and b_4 are essential nodes since the shortest path from s to any of them leads only via b_1. Nodes a_1 and b_1 are essential neighbors of s.

Theorem 1: The shortest path from a source node s toward any essential node routed via the same neighbor n leads only via nodes that are essential to s.

Theorem 2: Consider a shortest path $p = n_1, n_2, \ldots n_k$ of length k from a source node s to a node n_k. If a node $n_i \in p$ is non-essential then all nodes $n_j \in p \mid j \geq i$ are non-essential.

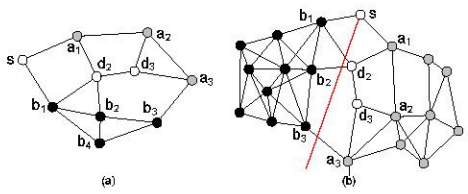

Fig. 2. (a) An example of a network used to demonstrate why a network field is divided by essential nodes. (b) An example of a network used to demonstrate why it is sufficient to select only a subset of all essential nodes for the process of building borders

A. Selection of Routing Neighbors

When a source node is building its FDR table, it has to determine how many zones will divide the network field ($|\mathcal{Z}^i|$) and select the corresponding routing neighbors. This process is performed locally and deterministically. Based on Theorem 1, we know that the shortest path to all essential nodes leads only via essential nodes.

Corollary 1: From Theorem 1, it follows that if there is an essential node e at distance α hops from a source node s, where $\alpha > 2$, there exists an essential node at distance two from s via which the shortest path leads to e.

Corollary 2: From Theorem 2, it follows that if there is a non-essential node n at distance β from a source node s, then there does not exists an essential node k at distance $d \geq \beta$ via which the shortest path leads via n.

First, we determine the number of routing zones and associate their routing neighbors. We describe this process using an example in Figure 2(a). In order to discriminate essential and non-essential nodes, s broadcasts a message to its neighbors requesting that they send the list of their neighbors. Node a_1 returns a list with nodes a_2 and d_2, and b_1 returns a list with nodes b_2, b_4, and d_2. Node s concludes that a_2 is only covered by node a_1 and therefore it is essential. Similarly, nodes b_2 and b_4 are essential, while node d_2 is covered by both neighbors. We denote d_2 a *don't-care* node. In this case, both neighbors are essential. Thus, s creates two routing zones with nodes a_1 and b_1 assigned to each zone. A source node selects all essential neighbors and a subset of non-essential neighbors as routing neighbors such that all nodes in the network are covered.

Figure 3 outlines the algorithm that identifies essential and non-essential neighbors, and selects routing neighbors. The goal of this algorithm is to select as few as possible non-essential neighbors that are assigned to routing zones. This way we heuristically aim at reducing the number of stored borders, i.e., the storage requirement, at each node. This problem can be reduced to the sequence covering problem, which is NP-hard.

SelectRoutingNeighbors
Input: source node s_i, set N_i of neighbors of s_i
Output: set of routing neighbors R_i
1. s_i requests lists $L(i,k)$ of neighbors from all $n_k \in T_i$
2. All $n_k \in T_i$ return $L(i,k)$; $L_i = L_i \cup L(i,k)$
3. **for each** node $x_l \in L_i$
4. **if** x_l appears in exactly one list $L(i,j)$
5. **then** x_l is an essential node, mark n_j as an essential neighbor
6. **add** all essential neighbors to R_i
7. Mark all nodes covered by neighbors already in R_i
8. **while** there are unmarked nodes
9. **add** to R_i a non-essential neighbor n_j that covers the largest number of unmarked nodes
10. mark all nodes covered by n_j

Fig. 3. Procedure that selects and assigns neighbors to routing zones.

B. Nodes Sufficient to Build a Border

The shape of a routing zone is dependent upon the positioning of the encompassed essential nodes. In the remainder of this manuscript, we refer to all nodes covered exclusively by a single routing neighbor as *essential nodes*. Nodes covered by more than one routing neighbor, are referred to as *don't-care* nodes. In both cases, for node n_k to *"cover"* node n_j from the perspective of a source node n_i, means that n_k is on the shortest path from n_i to n_j. If a source node identifies all essential nodes, it can identify all routing zones to achieve shortest-paths routing [2], which requires flooding. As this cost is prohibitive, we propose a more effective solution. In order to build borders between zones, it is not necessary for a source node to have knowledge of the entire network topology. Consider the example network in Figure 2(b). Source node s can build a border between two routing zones assigned to nodes a_1 and b_1 if it has the two lists of noedes: (*i*) a_2 and a_3 for neighbor a_1; and (*ii*) b_2 and b_3 for neighbor b_1.

We present FDR's Discovery protocol that identifies nodes necessary to build a border between two zones. The key idea behind this protocol is to send two messages along each side of the border. The messages carry the hop count from the source and a border side identifier (i.e., counterclockwise (CCW) or clockwise (CW)). This information enables nodes along the border to identify if they are essential or *don't-care* nodes from source's perspective. The messages propagate along the border until they reach an end of a network field. Then, essential nodes append themselves to the list of essential nodes on each side of the border and return the message to the essential node they got the message from. When two messages reach the source

node they contain the ordered lists of essential neighbors on each side of the border.

The pseudo-code for FDR's DISCOVERY protocol is presented in Figure 4. The pseudo-code is presented in the form of a recursive function that handles the requests for the list of essential nodes along the border. Each node can receive two types of messages from their neighbors: a *don't-care* announcement and a request for the list of essential nodes along the border. When a requests for the list is received, node x_i first determines if it is a *don't-care* node. In such a case, x_i announces to its neighbors that it is a *don't-care* node. Otherwise, x_i sorts its neighbors such that the closest node to the border is at the head of the sorted list. Since the border is not placed yet, x_i uses the reference point and a direction (CCW or CW) to determine the ordering. Then, x_i recursively requests the list of essential nodes from its neighbors in the sorted order. The bottom of this recursive procedure is when the end of the network field is reached.

DISCOVERY
Input: current node x_i, number of hops $\hat{h}$
Output: list of essential nodes L_i from x_i to the end of network field
1. **if** x_i is *don't-care*, x_i announces it is *don't-care* to all its neighbors
2. **return** empty L_i
3. **if** x_i is at the end of the network field
4. **return** $L_i = x_i$
5. Sort neighbors of x_i, $N_i = n_1 \dots n_k$ by distance to the border
6. **for each** neighbor $n_j \in N_i$
7. L_i = call DISCOVERY(n_j, $\hat{h}+1$)
8. **if** $L_i \neq$ empty prepend x_i to L_i, i.e., $L_i = x_i || L_i$
9. **return** L_i
10. **return** empty L_i

Fig. 4. Pseudo-code for FDR's Discovery protocol.

Theorem 3: DISCOVERY can yield suboptimal results in terms of shortest path routing and maintains node connectivity.

An important property of any routing scheme is its ability to forward messages without cyclic paths. Cyclic paths can result in buffer overflows, excessive energy bill, and dead-locks.

Theorem 4: FDR is an acyclic routing scheme.

BUILDBORDERS
Input: L_{CCW} and L_{CW} – lists of essential nodes,
$\hat{p}_{CCW}$ and $\hat{p}_{CW}$ – node pointers that walk L_{CCW} and L_{CW},
source node s
Output: C – list of border points
1. Set reference point $\hat{r} = s$
2. Set $\hat{p}_{CCW}$ and $\hat{p}_{CW}$ to the node closest to s in L_{CCW} and L_{CW}
3. Last valid pair of pointers $\{V_1, V_2\} = \{\hat{p}_{CCW}, \hat{p}_{CW}\}$
4. Compute $\hat{\theta} = \angle(\hat{p}_{CW}, \hat{r}, \hat{p}_{CCW})$
5. **repeat**
6. **if** $||\hat{p}_{CCW} - \hat{r}|| < ||\hat{p}_{CW} - \hat{r}||$ advance $\hat{p}_{CCW}$
7. **else** advance $\hat{p}_{CW}$
8. Compute $\theta = \angle(\hat{p}_{CW}, \hat{r}, \hat{p}_{CCW})$
9. **if** $\theta < 0$
10. **add** point c_i to C, c_i is on the line $\overline{V_1, V_2}$ and $||c_i - V_1|| = ||c_i - V_2||$
11. Place border between $\hat{r}$ and c_i; set $\hat{r} = c_i$;
12. $\hat{\theta} = \angle(V_1, \hat{r}, V_2)$; $\{\hat{p}_{CCW}, \hat{p}_{CW}\} = \{V_1, V_2\}$
13. **else if** $\theta < \hat{\theta}$
14. $\hat{\theta} = \theta$; $\{V_1, V_2\} = \{\hat{p}_{CCW}, \hat{p}_{CW}\}$
15. **until** both $\hat{p}_{CCW}$ and $\hat{p}_{CW}$ reach ends of L_{CCW} and L_{CW}

Fig. 5. Pseudo-code of the procedure that builds borders between two routing zones of a source node. Operator $||a - b||$ returns the Euclidean distance between two points a and b.

C. Building Borders

The source node initiates the procedure for building borders upon receipt of the two lists of essential nodes (CW and CCW from the border) from the associated pair of routing neighbors. The goal of this procedure is to create a chain of border line segments that separates two routing zones such that node motion is maximally tolerated. This is accomplished by placing the border equidistantly from closest essential nodes in each zone.

The pseudo-code of this procedure is shown in Figure 5. The procedure does not attempt to insert the minimal number of border points in order to address the (*vii*) criterion. We have opted for a strategy where a border point is inserted equidistantly between two conflicting pointers. This enables less frequent updates of borders since changes in node positions have the least effect on borders. As most of the borders are composed of only a single border point, the overall increase in the average routing table size is negligible.

D. Routing Table Inheritance

Messages sent throughout the network with a purpose to discover its topology and establish routing tables at specific nodes, are considered a routing overhead. In order to reduce this overhead, we propose an INHERITANCE protocol that builds a routing table at a single node by analyzing the already computed routing tables of the node's neighbors. In most cases, routing zones of two routing neighbors overlap. The overlapping area contains nodes that can be routed via either of the two routing neighbors. It is necessary to determine how to divide that area to preserve near-optimal shortest path routing in the expected case. The proposed INHERITANCE protocol estimates the required hop counts along the intersecting borders to determine equidistant points between them. We consider these points as border points of the new border. Thus, the border is constructed as an estimate based upon neighbor's routing. Pseudo-code for the INHERITANCE protocol is presented in Figure 6.

INHERITANCE
Input: C_{CCW}, C_{CW} – lists of border points with the hop count information for each point, source node s
Output: C – list of points for the inherited border
1. **for each** border line $l = \overline{c_i, c_{i+1}}$ in C_{CCW} and C_{CW}
2. insert $\phi_{i+1} - \phi_i - 1 = P - 1$ pseudo border points $p_0 = c_i, p_1 \dots p_{P-1}, p_P = c_{i+1}$ such that $(\forall j) p_j \in l$ and $||p_{j+1} - p_j|| = ||p_j - p_{j-1}||$
3. **for each** p_j
4. compute its hop count estimate: ϕ_j
5. **for each** border point $c_i \in C_{CCW} \cup C_{CW}$
6. **if** $(\exists p_j)\phi_j = \phi_i$ and p_j is at opposing border to c_i
7. **add** point t to C where t is on $\overline{c_i, p_j}$ and $||c_i - t|| = ||p_j - t||$

Fig. 6. Pseudo-code of the procedure that builds a border between routing zones of two adjacent routing neighbors of a source node.

We analyze two specific situations. When neighbors' two border zones do not overlap, then INHERITANCE uses the procedure BUILDBORDERS from Figure 5 to build a new border in-between the two non-intersecting borders. The two lists of border points that correspond to the non-intersecting borders are fed as the L_{CCW} and L_{CW} input to BUILDBORDERS. Finally, a node that has used INHERITANCE to derive its routing table, can fine-tune its borders for destinations that are frequently contacted and are located near the border. The fine-tuning can be performed by sending a test message via each of the candidate routing neighbors, learning the hop count to them, and readjusting the border.

Theorem 5: INHERITANCE can yield sub-optimal results in terms of shortest path routing.

Theorem 6: INHERITANCE preserves connectivity.

Theorem 7: INHERITANCE preserves acyclic routing.

E. Synchronizing the Initialization

We present an efficient protocol, SYNCHINIT, for distributed and localized network initialization that combines routing table creation via the DISCOVERY and INHERITANCE protocols. The prerequisite condition for applying INHERITANCE is that routing neighbors of the source node have already initialized their routing tables. We can assess the following optimization goal: knowing the network topology, select minimal number of nodes whose routing tables are initialized via DISCOVERY, such that remaining nodes in the network can initialize their routing tables via INHERITANCE. In a centralized case, this problem can be reduced to the MINIMUM SET COVER problem, which is NP-hard. To address the distributed variant of this problem, we propose an effective heuristic with emphasis on protocol simplicity.

The key idea behind SYNCHINIT is to overlay the network field with a regular grid and initialize nodes closest to grid intersections via DISCOVERY. The remaining nodes are then initialized via INHERITANCE if their prerequisite conditions are satisfied. Next, if there still exist uninitialized nodes, SYNCHINIT increases the grid density and

repeats the previous two steps. These two steps can be iterated until all nodes are initialized. Alternatively, SYNCHINIT can repeat fixed number of iterations and then force all uninitialized nodes to perform DISCOVERY to complete network initialization.

SYNCHINIT
Input: Node n, comm. range r, network field $\mathcal{F}$
Output: Initialized routing table of n
1. Overlay $\mathcal{F}$ using a regular grid $\mathcal{G}$; the shortest distance between two points in $\mathcal{G}$ is $2r$
2. Find the closest grid point g
3. **repeat** R times
4. **if** n is the closest uninitialized node to g **and** $||n-g|| < r/2$
5. **return** DISCOVERY(n)
6. **repeat** Q times
7. **if** routing neighbors of n are initialized
8. **return** INHERITANCE(n)
9. Double the grid density
10. **return** DISCOVERY(n)

Fig. 7. Pseudo-code for distributed initialization of a routing table. In step **4**, if there is a tie in terms of distance, nodes are ordered clockwise north-first to break the tie. In our experiments, we limited $R = 3$ and $Q = 2$.

Pseudo-code for this protocol is illustrated in Figure 7. We evaluate two key trade-offs related to SYNCHINIT. First, a denser grid causes more nodes to initiate DISCOVERY. Then, network initialization converges faster at the expense of sending more messages. Second, we initiate INHERITANCE Q times for each iteration of SYNCHINIT (steps **6** through **8**). Nodes that have built routing tables after DISCOVERY can use their routing table to initiate INHERITANCE in other nodes. However, this is not feasible if nodes have a mutual relationship of being routing neighbors to each other. Such nodes cannot use INHERITANCE to build their tables.

V. SUPPORT FOR NETWORK DYNAMICS

In this section, we propose protocols that can efficiently cope with the management of nodes' routing tables in the case of a dynamic network topology. We do not bound the space of possible changes in the network. FDR supports introduction of new and failure of existing nodes and/or obstacles to communication in the network field. The two atomic events that can model any of these cases, are *appearance* and *disappearance* of a node in/from the communication range of another node. We denote them as *motion* events. When a motion event occurs, certain nodes may become detached from all but the neighboring nodes in the network because their routing tables are non-existent or invalid. This condition may last until the node seizes its motion and some or all routing tables of its neighboring nodes are updated.

From the perspective of a specific source node, most of the changes in the network do not have any impact on its routing table. If a motion event occurs far from its routing table's borders, usually it does not affect source's routing table. Changes in one part of the network have a smaller expected effect on nodes located far from the place where the motion event occurred. This property enables FDR to be a highly scalable routing scheme when dealing with network dynamics.

Node appearance is a motion event after which two nodes are able to directly communicate. This event can occur due to introduction of a new node in the network, node motion, or motion of a communication obstacle. The objective for the protocol that maintains nodes' routing tables is to identify whether any tables within the network require an update of their routing rules and if yes, to perform the updates in the least expensive fashion.

When two nodes, n and m, establish communication upon a motion event, each of them executes the MOTIONUPDATE procedure outlined in Figure 8. After learning each others' geographical locations, each node computes its list of routing neighbors as described in Figure 3. Note that n includes m in its list of routing neighbors *only* if m is an essential neighbor to n. In case m is a routing neighbor, it changes the network topology for n sufficiently so that n needs to update its table via the DISCOVERY protocol. If there is any change to its routing table borders, n must propagate these changes to its neighbors. Thus, n sends its routing table to all its neighbors.

A node n that receives the propagation packet from its neighbor m, executes the MOTIONPROPAGATE procedure. It ignores the package if m is not its routing neighbor or if the received routing table actually does no affect any of the borders in the existing routing table of n. Otherwise, it recomputes its routing table based on the INHERITANCE protocol. Note that only borders affected by the propagation package are recomputed. If there are any changes to the borders of the routing table of n, node n must propagate these changes to its neighbors. The main property of the updating protocols is that DISCOVERY is performed only by the nodes directly affected by the motion event. In case there are any changes to the routing tables, they are propagated using the localized INHERITANCE protocol. In addition, the propagation typically occurs in the immediate neighborhood of the motion event. Only events that profoundly change the topology of the network initiate network-wide propagation.

Node disappearance is an event dual in nature to node appearance. The update procedure for this event is equivalent to the protocol presented in Figure 8 with two key differences. First, it is triggered by the disappearance of a neighbor m from the communication range of a given source node n. Second, during step **2** of MOTIONUPDATE, n initially tries to find another neighbor m' that can replace m and preserve the existing routing table borders. If this cannot be achieved, n recreates the list of routing neighbors with an aim to preserve as much as possible of the borders from the existing routing table. This objective minimizes the number of neighbors that propagate their routing table changes.

MOTIONUPDATE
Input: Source node n, existing routing table τ_0 of n, node m appears in its communication range
Output: Routing table of n
1. Exchange geographic location data with m
2. $L =$ SELECTROUTINGNEIGHBORS(n)
3. **if** $m \in L$
4. Routing table $\tau =$ DISCOVERY(n)
5. **if** $\tau_0 \neq \tau$
6. **for each** neighbor p to n
7. $e_p =$ PRNG(), send $\{\tau, e_p\}$ to p
8. **return** τ
9. **return** τ_0

MOTIONPROPAGATE
Input: Motion event e, source node n, its neighbor m, and their routing tables τ_n^0 and τ_m respectively
Output: Routing table of n
1. Node n receives $\{\tau_m, e\}$ from m
2. **if** m is a routing neighbor to n **and** n has not yet updated its routing table due to e **and** τ_m affects at least one of the borders in τ_n^0
3. $\tau_n^1 =$ INHERITANCE(n)
4. **if** $\tau_n^1 \neq \tau_n^0$ **then** send $\{\tau_n^1, e\}$ to all neighbors
5. **return** τ_n^1
6. **return** τ_n^0

Fig. 8. Pseudo-code for routing table update upon a motion event. Function PRNG() returns a pseudo-random number.

VI. EXPERIMENTAL RESULTS

In order to evaluate the performance of the generic FDR platform, we conducted several experiments. First, we created a network field with four obstacles in a realistic setting, i.e., a skewed distribution of nodes' placement in the field. Then, we randomly generated three network instances with 200, 500, and 1000 nodes and measured the percentage of messages that reached destination for each node-to-node coupling in case GPSR was used as a routing mechanism [5]. Only 79%, 88.6%, and 90.3% of messages respectively were delivered, while the remainder had to terminate their path search due to exceeded TTL. Most applications that require reliability pose a strong demand for alternate routing mechanisms that guarantee delivery.

Figure 9 (left and center) illustrates histograms of path lengths for all pairs of nodes for two randomly generated networks of $\{N, r\} = \{200, 0.13\}, \{400, 0.088\}$. Range r was chosen so that the expected number of neighbors for each node is approximately the same; we aimed at networks of similar density and different area coverage. We compared the all-shortest-paths computed via the Dijkstra algorithm with paths computed via the DISCOVERY and

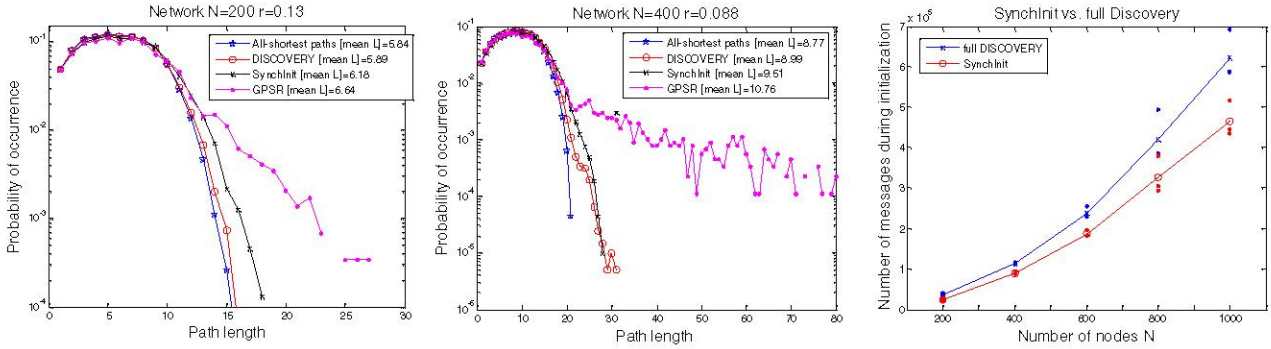

Fig. 9. Probability distribution for path length in a randomly generated network $N = 200, r = 0.13$ (left) and $N = 400, r = 0.088$ (center) in a unit-square field. Number of messages sent during initialization using DISCOVERY only and using SYNCHINIT for three network instances for each of the following network types: $\{N, r\} = \{200, 0.13\}, \{400, 0.088\}, \{600, 0.071\}, \{800, 0.061\}, \{1000, 0.06\}$. We selected the communication range r for each network so that the expected number of neighbors for each node is approximately equivalent.

SYNCHINIT protocols. Also, we computed path-lengths using the GPSR protocol *excluding* connections that were not established. One can observe that DISCOVERY found optimal paths in nearly all cases (overhead of less than 1% and 2.5%) and SYNCHINIT produced overhead of 5.8% and 8.4% compared to GPSR's overhead of 9% and 23% (with all infeasible paths excluded) for the network instances of 200 and 400 nodes respectively.

Figure 9 (right plot) illustrates the number of messages, M, exchanged among all nodes during network initialization. This is the full overhead of establishing connectivity via FDR. Considered networks are itemized in the caption of Figure 9. Two different datasets are plotted, M for initialization via DISCOVERY only and via SYNCHINIT. In almost all cases, we recorded improvement in traffic that was nearly constant within 20-25%. For denser networks, this improvement increased, e.g., in our experiments we recorded the largest improvement in excess of 60% for $\{N, r\} = \{1000, 0.13\}$. Note that the number of messages is comparable to network "flooding;" however, FDR has several important improvements with respect to other "flooding" schemes. First, in FDR $M \sim \mathcal{O}(N\sqrt{N})$ vs. $\mathcal{O}(N^2)$ for true "flooding." Next, individual nodes do not have to compute the shortest paths upon learning network's topology. This greatly reduces the complexity and memory requirement of individual nodes. In addition, FDR's routing tables require nearly minimal storage which scales well with network size [2]. Finally, FDR supports mobility mostly via the INHERITANCE primitive which greatly reduces traffic for routing tables' maintenance once connectivity is established via SYNCHINIT.

For experiments with motion events, we recorded the trail of messages sent throughout a network in order to propagate information about changes in network topology. We used a uniform distribution to determine a direction of each motion event. We used an exponentially decreasing distribution function to determine the length of each move. For each network type, we simulated 25 000 motion events over 5 different network instances. Each motion event can affect a routing table of a number of nodes, and one or more routing borders within each table. The average number of modified tables ranges from 12.3 for networks with $\{N, r\} = \{200, 0.130\}$ to 17.8 for $\{N, r\} = \{1000, 0.060\}$.

The likelihood of a node launching the DISCOVERY protocol due to a motion event ranges from 45% for networks with 200 nodes, to 10% for networks with 1000 nodes. Figure 10 presents the overall improvement of the cost (expressed in terms of the number exchanged messages necessary for a network update) that is the result of a use of the combination of DISCOVERY and INHERITANCE protocols, compared to the incurred cost if only the DISCOVERY protocol is used. The combination of the protocols significantly reduces communication requirements, while maintaining the property that *all* tables have updated routes toward all nodes in the network.

VII. CONCLUSION

In summary, the detailed presentation of the FDR framework in this paper along with the preliminary experimental results, showcases a multi-hop protocol that addresses efficiently the criteria (*i-vii*) for WAHNs. Compared to GPSR, traffic overhead due to periodic SYNCHINITs to optimize path lengths can be negligible in WAHNs where nodes exchange large amounts of data (e.g., audio/video, sensor data) and motion events happen infrequently. Compared to other proactive schemes, FDR offers support for node motion at low cost in overhead traffic and requires low computational resources for nodes. This can be particularly applicable to networks of sensors.

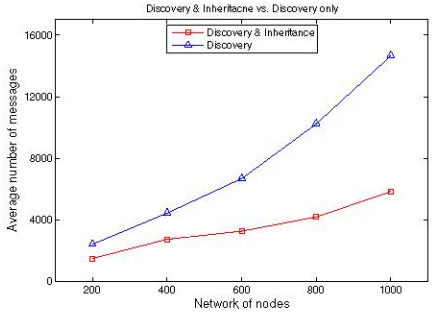

Fig. 10. Comparison between a system using only the DISCOVERY protocol and a system that combines the DISCOVERY and INHERITANCE protocols.

REFERENCES

[1] E. Royer and C. Toh, "A review of current routing protocols for ad-hoc mobile wireless networks," *IEEE Personal Communications*, 1999.

[2] M. Drinić, D. Kirovski, and M. Potkonjak, "Model-based compression in wireless ad hoc networks," *ACM SenSys*, 2003.

[3] B. Bellur and R. G. Ogier, "A reliable, efficient topology broadcast protocol for dynamic networks," *INFOCOM The Conference on Computer Communications*, vol. 1, pp. 178–86, 1999.

[4] C. Perkins and E. M. Royer, "Ad hoc on demand distance vector (AODV) routing," *IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, Feb. 1999.

[5] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," *Mobile Computing and Networking*, pp. 243–54, 2000.

[6] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Practical and robust geographic routing in wireless networks," *Embedded Networked Sensor Systems*, pp. 295–6, 2004.

[7] P. Samar, M. R. Pearlman, and Z. J. Haas, "Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 4, pp. 595–608, 2004.

[8] N. Nikaein and C. Bonnet, "Ddr: distributed dynamic routing algorithm for mobile ad hoc networks," *Mobile Ad Hoc Networking & Computing*, pp. 19–27, 2000.

[9] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," *Conference On Embedded Networked Sensor Systems*, pp. 14–27, 2003.

[10] J. Li, J. Jannotti, D. S. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad-hoc routing," *ACM Mobicom*, pp. 120–30, 2000.