

Model-Based Compression in Wireless Ad Hoc Networks

Milenko Drinić[†], Darko Kirovski[†], and Miodrag Potkonjak[‡]

[†] Microsoft Research, One Microsoft Way, Redmond, WA

[‡] Computer Science Department, University of California, Los Angeles, CA

{mdrinic,darkok}@microsoft.com, miodrag@cs.ucla.edu

ABSTRACT

We present a technique for compression of shortest paths routing tables for wireless ad hoc networks. The main characteristic of such networks is that geographic location of nodes determines network topology. As opposed to encoding individual node locations, at each node our approach groups the remaining nodes in the network into regions. All shortest paths to nodes in a specific region are routed via the same neighboring node. In this paper, we propose an algorithm for dividing a network field into distinct regions to minimize routing table size while guaranteeing shortest path routes. We show that this problem is NP-hard, propose a heuristic to find efficient solutions, and empirically demonstrate the resulting system performance from the perspective of compression ratio and scalability. In our experiments, routing tables compressed using this technique, require 88.9% to 97.9% less storage than uncompressed tables.

In order to achieve energy efficient routing, we propose an augmentation to the original routing mechanism that enables load balancing flexibility along with guaranteed shortest path routing at the expense of larger routing tables. Preliminary experiments estimate 10% lifetime extension of network nodes with a tradeoff of an increase in the size of routing tables. Finally, we propose a compression technique that aims at representing trajectories in a sensing network in a compact manner. This approach relies on trajectory prediction using three weighted Markov models, a local, regional and global one, all of them with context-length equal to one. Finally, we discuss a range of possible applications that rely on the developed prediction and routing models.

Categories and Subject Descriptors

C.2.2 [Computer-communication networks]: Network protocols, Routing protocols; E.4 [Coding and information theory]: Data compaction and compression

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SenSys'03, November 5–7, 2003, Los Angeles, California, USA.

Copyright 2003 ACM 1-58113-707-9/03/0011 ...\$5.00.

General Terms

Algorithms

Keywords

Sensor networks, routing protocols, compression, routing tables, trajectory, modeling.

1. INTRODUCTION

Ad hoc wireless networks are commonly envisioned as the first link in the chain of actions sensing - computation - actuating where computing machines impact reality. The wide range of applicability of ad hoc wireless sensor networks stems from their ease of setup, pervasive presence, and ability to monitor areas without modifications. In a common model of sensor networks [11, 12, 14], they are composed of a number of small size devices with sensing, processing, and communication capabilities. Each sensing node in the network monitors phenomena, objects, and events. Collected data is processed to the extent of computational power of each node, and distributed throughout the network via the node's radio, audio, or optical connection [4, 7]. Each node is also routing messages to and from nodes that are within its radio communication range. The limitations on power consumption imposed by the power supply of each nodes results in a node to node communication range that is typically small compared to the total network coverage area. Limited communication range also provides better utilization of the communication bandwidth because nodes in different areas of the network are able to communicate simultaneously without interference.

The largest portion of energy in a sensing node is consumed while communicating, which includes both reception and transmissions of data packets [3, 20]. Routing strategies play an important role in the minimization of the number of transmitted messages. A minimal number of message transmissions is obtained if the communication path is composed of minimal number of hops.

1.1 Compression of Routing Tables

Wireless multi-hop routing protocols encounter scalability issues when network size increases. The size of a routing table for networks with a large number of nodes is limited to the available memory in a sensing node. It also prolongs the search for a particular entry in the table. At the same time, maintenance of large tables requires intense communication and coordination among nodes. In this paper we present a novel, compact, and scalable representation of shortest

path routing tables based on modeling geographical data in network field. Geographical data play an important role in wireless ad hoc networks because nodes' positioning determines network topology.

We model a network field as a set of non-overlapping regions from the perspective of each node. Consider an example of a wireless network with 100 nodes in Figure 1. In the example we consider node A as a source node. Node B is neighbor of node A since it is within the communication range of node A . We assign non-overlapping region $R_{A \rightarrow B}$ of source node A to node B . Region $R_{A \rightarrow B}$ is a region between lines L_1 and L_2 in counter clockwise (CCW) direction. Shortest paths from the source node to all nodes in this region have a common characteristic that their first hop is node B . For message routing, it is sufficient to identify to which region the destination node belongs and to forward a message to the assigned neighbor.

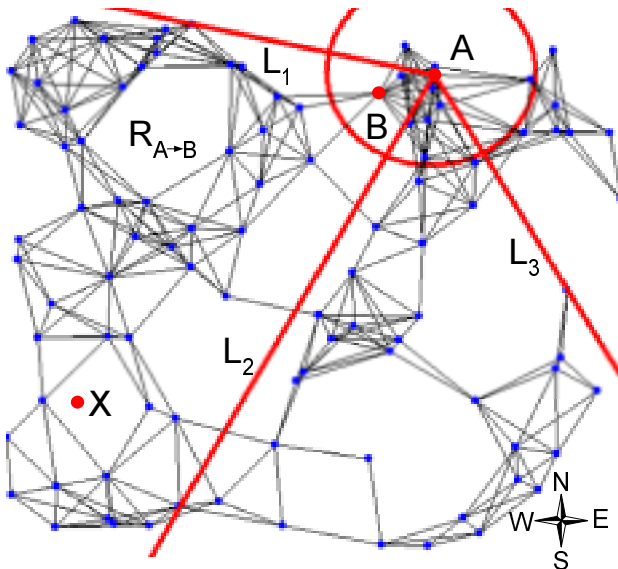


Figure 1: An example of the division of a network field into regions from the perspective of source node A . The network shown has 100 nodes which are distributed over a square field of unit size. Node distribution is random and uniform. Communication range of each node is 0.17 relative units. For node A , its communication range is shown as a circle centered in A .

In the example shown in Figure 1, the shortest routing table of source node A can be represented with only three points in the space. Each of the points uniquely represent borders L_1 , L_2 , and L_3 that are shared between the regions. Three regions are assigned to three neighbors of source node A . The procedure of sending data packets from node is as follows. First, node A identifies the region where the destination is positioned. Then it transmits the message to the neighbor that is assigned to the identified region. For a successful message transmission the destination point (point X in Figure 1) is not restricted to nodes in the network. It can be any point in the network field. In the example in Figure 1, the source node is node A while the destination point is denoted as X . Source node A identifies region $R_{A \rightarrow B}$. Then it transmits the coordinates of point X to its neighbor B .

Node B repeats the procedure acting as a current source node. Finally, the message is delivered when the current source node and the destination are within the communication range.

Our representation of the shortest path routing table is multiple times smaller than the full shortest path routing table. Depending on the network size and density, the obtained compression ratios range from 2.1% to 11.1%, i.e. memory requirements are reduced from 88.9% to 97.9%. The compression ratio is defined as the ratio between the compressed representation of routing table, and the original representation containing the same amount of information.

We have propose an augmentation to our routing algorithm in order to enable load balancing. From a source node point of view, a number of nodes has a shortest path via multiple neighboring nodes. Such nodes are positioned along the region borders. We have modified our algorithm such that each region encompass all nodes that have the shortest path via assigned neighbor. Consequently, routing regions overlap and borders are not shared. A source node now has the flexibility to choose the first hop node with higher remaining energy if the destination belongs to more than one region. This way, the load of message routing is balanced throughout the network. With the modification, preliminary experiments estimate 10% time extension of the first node failure because of low energy. The load balancing is enabled with an increase of the size of routing tables. However, memory requirements remain multiple times smaller than for a full, uncompressed routing table.

1.2 Compression of Trajectories

Tracking objects is a standard application in sensor networks that is usually associated with large number of long trajectories that require significant storage. Trajectory information is typically represented as a set of coordinates in two or three dimensional space. Trajectories' consecutive coordinates are spatially correlated since objects are traveling in a certain direction. However, general purpose compressors are unable to capture these correlations because: *a.* data representation hides them, and *b.* prediction mechanisms used in compressors identify the exact pattern repetition. If the input to a general purpose compressor is in the form of absolute coordinates, the compressor is unable to extract direction information that is crucial for compressing trajectories (reason *a.*). Frequent small changes in directions are common in trajectory data and they are mispredicted when the exact patterns are identified (reason *b.*).

We present a novel model-based compression technique that improves data modeling of trajectories. Our technique consist of two steps:

- (i) Data transformation from an absolute coordinates representation to a representation with relative positions.
- (ii) Compression of the relative positions based on the local, regional, and global trajectory history.

The first step converts absolute coordinates into a representation where each position of a point is represented as a difference from the previous position. This relative representation is significantly smaller than its absolute position counterpart. The first step exposes the correlations between subsequent positions that are not visible to compressors in absolute point representation. This enables capturing of spatial correlations, which is performed in the second

step. For the second step, we have developed a compressor that specifically targets trajectory data. The compressor is adapted to capture idiosyncrasies of trajectory data by using local, regional, and global history. Three histories are represented as weighted Markov models of order 0 and encoded with an arithmetic coder [19]. Our compression technique yields compression ratios in the range from 1.2% to 3.2% for the given data set. For the same input, an LZ based compressor yields compression ratio in the range from 15.9% to 31.7%

The straightforward application of our technique is for minimization of storage requirements. One of the features of our model-based compression technique is that it exposes entropy of trajectories [19]. This can be used for applications that predict objects' positions. Better compression ratio indicates better prediction of symbols in the input stream. In this case, the symbols are objects positions. Alternatively, our technique can be used for profiling of objects' behaviors. For example, artificially guided and remote control guided vehicles have significantly different entropies that can be differentiated by our technique.

2. RELATED WORK

Routing protocols can be classified in most cases as *proactive* or as *reactive*. Proactive protocols continuously evaluate network topology and update routing tables. When a packet is ready for transmission, the forwarding path is already known and can be immediately used. Earlier examples of proactive protocols are called Distance Vector protocols, which are based on the Bellman-Ford algorithm [6]. Modifications of the basic algorithm address the issues of convergence and excessive traffic [9, 17]. Link State protocols converge faster at the expense of increased control traffic [12]. By combining Distance Vector and Link State protocols, the Wireless Routing Protocol eliminates the "counting-to-infinity" problem and reduces the control traffic [15].

In contrast to proactive protocols, reactive protocols create packet forwarding routes upon request. When a source node is ready to transmit a packet, it activates a route discovery mechanism to find the path to the destination. Route discovery is based on the exchange of query and reply messages, where the query is flooded through the network. In the Temporally Ordered Routing Algorithm (TORA) the reply is flooded back to the source in a controlled manner in the form of a directed acyclic graph with the root in the destination [16]. Dynamic Source Routing (DSR) [13] and Ad Hoc On Demand Distance Vector (AODV) [18] protocols use the route constructed during querying to reply to the source.

Alternative approaches to proactive and reactive protocols are hybrid protocols such as Zone Routing Protocols (ZRP) [11], and localized algorithms for route discovery such as Greedy Perimeter Stateless Routing (GPSR) [14]. These two approaches rely on the geographical information of nodes to make a decision about the packet forwarding route.

Recently, a number of routing algorithms in wireless ad hoc networks have been developed for reducing energy consumption. *Span* is a distributed and randomized algorithm where a node makes local decisions on whether to turn off their transmitters (sleep mode) or to join forwarding backbones of the network [3]. Geographical Adaptive Fidelity (GAF) identifies nodes that are equivalent from a routing

standpoint, and decides which nodes can go to sleep mode. An alternative approach for energy conservation was proposed by Chang and Tassiulas where they explore selection of message routes such that energy consumption is balanced among nodes in proportion to their energy reserves [2].

3. FIELD DIVISION ROUTING

Our algorithm leverages on high correlation of geographical information and shortest path routing in a sensor network to derive routing tables of minimal size. Node positions in a wireless communication environment determines network topology. Consequently, there is a high correlation between geographical positions of nodes and routing paths in a wireless ad hoc network. Our compression scheme is based on modeling of network topology and its minimal representation. In contrast, standard compression schemes use computationally intensive algorithms in an attempt to efficiently encode information.

Let us consider again the example shown in Figure 1. In terms of geographical positions, node *B* lies approximately westward from node *A*. These two nodes are within communication range of each other. Node *B* is also the node that is positioned the furthest west of all neighbors of *A*. Because of the geographical correlations, we assume that the shortest path for most of the nodes westward from node *A* will go via node *B*. In sufficiently dense and connected networks, this assumption holds. In networks that have areas scarcely covered by nodes or not covered at all by any node, this assumption is not necessarily correct.

Consider the example shown in Figure 2 with eight nodes where single hop coverage areas for nodes *A* and *C* are shown as dotted circles centered in *A* and *C*. The shortest paths from *A* to nodes *F* and *G* lead via *B*. The path lengths are 3 and 4 hops respectively. If we draw our conclusion only on geographical position of nodes *F* and *G*, routing should proceed via node *C*. This would result in longer paths of 4 and 5 hops respectively assuming that *C* forward it to *B* instead to node *H*. From the perspective of geographical correlation, it can be noticed that *F* and *G* are more correlated to node *B* than to node *C* because of their geographical correlations with nodes *E* and *D*.

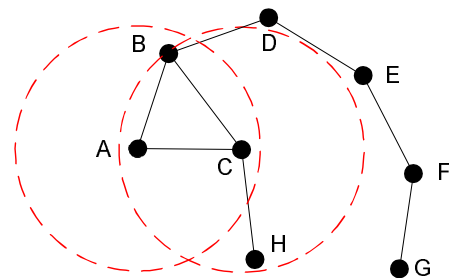


Figure 2: An example of an network topology where the shortest path between two nodes (*A* and *G* or *A* and *F*) does not lead via the node in the same direction (node *C*) as the associated neighbor (node *B*).

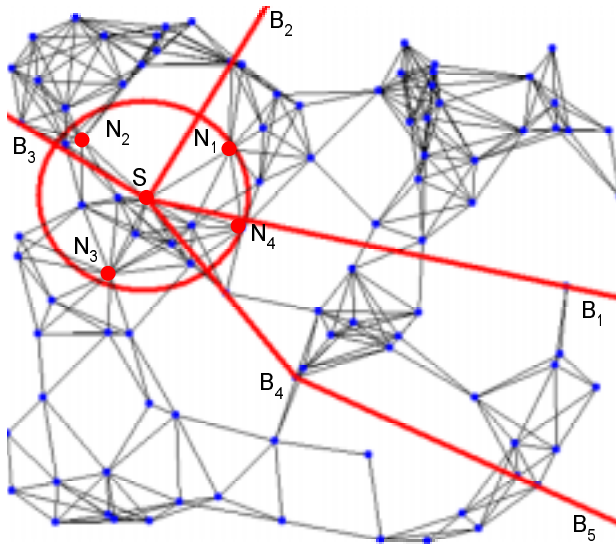
Geographical correlations can be described in the form of regions that divide the sensor network from the perspective of a source node. Each region is affiliated with one of the neighbors of the source node. All nodes in the region

have the shortest path from the source node via the assigned neighbor. The implementation of our algorithm assumes that regions can be open or closed. An open region is defined as such a region where the first and the last points are different, i.e. the lines of the region do not create an enclosed area. Borders of regions are straight lines. Higher order polynomials (curve lines) that could describe region borders offer more flexibility but they introduce additional complexity to the algorithm, the description of routing tables, and in determining a node location within a region.

Our shortest path routing table consists of:

- A *List of Neighbors*, via which all the nodes in the network are reached by shortest paths;
- A *List of Border Points* for each region. Each list is affiliated with one of the neighboring nodes from the *List of Neighbors*.

An example of a routing table with its corresponding source node is depicted in Figure 3. The nodes are listed in CCW order. The *List of Neighbors* is a simple description of a circular adjacency of the regions affiliated with the nodes from the list, if all regions are open such as in the example in Figure 3. The starting *Border Point* for each region is the position of the source node. This information is implicit and it is not listed in the table. If the *List of Border Points* ends with the position of the source node, such region is closed. In such a case additional information is inserted in the table describing the adjacency of regions. During our experiments, we have encountered closed regions seldomly, to the extent that their effect on the complexity of the table at average can be entirely neglected.



Shortest path routing table of node S

List of Neighbors	N_1	N_2	N_3	N_4
List of Border Points	B_1	B_2	B_3	B_4 B_5

Figure 3: An example of a model-based compressed shortest path routing table for source node S in a network of consisting of 100 nodes.

3.1 Essential Nodes, Essential Neighbors, and "Don't-Care" Nodes

Before we describe our algorithm for building the routing table, we introduce the following definitions and lemmas that enable formal description of the algorithm. First, we denote the procedure that calculates the length of the shortest path between nodes A and B as $SPL(A, B)$. If $SPL(A, B) = 1$, nodes A and B are neighboring nodes.

DEFINITION 1. *Essential node E of source node S has a property that $SPL(S, E) = SPL(N, E) + 1$ where node N is such that $SPL(S, N) = 1$, and that does not \exists node M with $SPL(S, M) = 1$ such that $SPL(M, E) = SPL(N, E)$. We say that node E is an essential node of node S via N .*

DEFINITION 2. *Essential neighbor N of source node S is such a node where $SPL(S, N) = 1$ and \exists essential node E such that $SPL(S, E) = SPL(S, N) + 1$.*

DEFINITION 3. *Don't-care node D of source node S is a node for which \exists nodes M and N ($M \neq N$) such that $SPL(S, D) = SPL(M, D) + 1 = SPL(N, D) + 1$, and that $SPL(S, M) = SPL(S, N) = 1$.*

Figure 4 illustrates examples of essential nodes, essential neighbors, and don't-care nodes.

Based on Definitions 1 and 2 we prove the following lemmas about the existence of an essential neighbor and path to essential nodes. The lemmas are used for the proof of the correctness of the routing protocol and the optimality of the algorithm for building borders.

LEMMA 1. *Neighbor N of source node S is its essential neighbor if and only if \exists node M such that $SPL(N, M) = 1$, and node M is an essential node of S .*

Proof. By contradiction. Let us assume that the lemma claim is incorrect. That would mean node N is an essential neighbor of S , and all of its neighbors are non-essential from the perspective of S . More formally, $(\forall K)$ such that $SPL(N, K) = 1$, $(\exists M)$ such that $SPL(S, M) = 1$ and $SPL(M, K) = 1$ or $SPL(S, K) = 1$. By Definition 1, essential node E of S is such a node where there is only one neighbor of node S via which node E can be reached in the minimal number of hops. By the counterclaim to the lemma, $(\forall E) SPL(N, E) > 1$. Let us assume that K is the next hop from N toward E , i.e. $SPL(N, E) = SPL(K, E) + 1$. Such a node must exist since N and E are connected. By the counterclaim of the theorem, K is not an essential node to S , which means there is an alternate route to K such that $SPL(S, K) \leq 2$. This means that there is an alternate route from S to E , which contradicts the fact that E is essential node. \square

LEMMA 2. *Shortest path from node S to its essential node E is routed only via nodes that are essential from the perspective of S via the same essential neighbor N .*

Proof. By induction. Directly from Lemma 1 it follows that all essential nodes of S with a distance of two hops routed via essential neighbor N are routed via the same essential neighbor. We identify this fact as a base case for our proof. Let us assume that there exists a node L_k that is essential node of S via N , and $SPL(S, L_k) = k$. Let us consider

neighboring node L_{k+1} of L_k with $SPL(S, L_{k+1}) = k + 1$, which is essential node of S . The shortest path from S to L_{k+1} can be routed via node L_k . Simultaneously, node L_{k+1} cannot be routed via another node L'_k with the shortest path of length $k + 1$ such that L'_k is not an essential node of S via N . Otherwise, node L_{k+1} would not be essential node of S , or its shortest path would not be $k + 1$, which contradicts the initial assumption. It follows that L_{k+1} is routed via an essential node that is routed via the same essential neighbor of S as L_{k+1} . \square

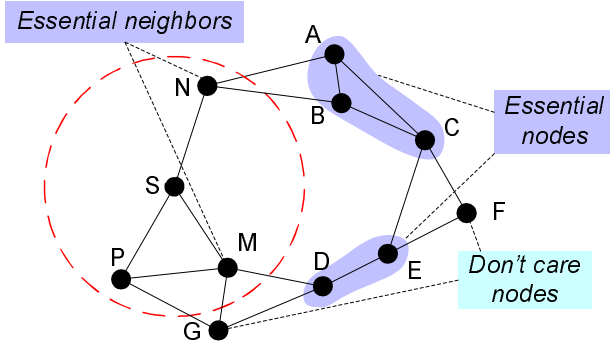


Figure 4: An example of essential nodes, essential neighbors, and *don't-care* nodes. Nodes A, B, and C are essential nodes of node S via node N while nodes D and E are essential nodes of S via node M. Nodes F and G are *don't-care* nodes of S.

Figure 4 illustrates the relationship between a source node, essential nodes, essential neighbors, and *don't-care* nodes. Node N is essential neighbor of node S because of essential nodes A, B, and C, while node M is an essential neighbor of S because of essential nodes D and E. There are two shortest paths from source node to node F via different neighbors of S , which, by Definition 3 implies that node F is a *don't-care* node of S . Similarly, node G is also a *don't-care* node of S . Node P , which is a neighbor of S , is not an essential neighbor because it has no direct connection to an essential node (Lemma 1). All essential nodes of S via N compose a connected subgraph with its root in N as suggested by Lemma 2. Note that Lemmas 1 and 2, and Definition 1 do not imply that there is only a single path from source to an essential node. There can be multiple shortest paths from source to the same essential node, and all of them are leading via a single essential neighbor. In Figure 4, node C has two shortest paths from node S of length three, both leading via essential node N .

3.2 Compression Algorithm of Field Division Shortest Path Routing Table

In this section, we describe an algorithm for inserting borders between regions in a compressed form for field division routing. We assume that initially there exists information about the position of nodes inside the network field. We also assume that the initial regions for each node are calculated at a location with significantly more computational power than a sensing node (such as a base station), because of the high demand for computational resources and communication between nodes. The distribution of the initial routing tables is highly effective because of their minimal size. Alternatively, all initial computations could be per-

formed distributively, on sensing nodes. Initially an instance of Distance Vector or Link State protocols can be used to discover network topology.

Pseudo code of the compression field division shortest path routing table is shown in Figure 5 a). The first step of our algorithm is the calculation of all-pairs shortest paths in the network by using Johnson's algorithm [6]. Then, as the second step, we select neighbors for each node in the network. Each of the selected neighbors will be affiliated with a region. When neighbors and nodes are selected, the third step is the border insertion for each pair of selected neighbors. Finally, the regions are adjusted for possible closed regions.

In order to formally abstract the neighbor selection problem, we derive a neighbor coverage graph of source node S (NCG_S) from network topology. NCG_S has two sets of nodes: (i) neighbor nodes of S , and (ii) the remaining nodes from the network excluding S . For each pairs of nodes (N, M) where N is from set (i) and M is from set (ii) of NCG_S we add an edge between them if there is a shortest path from S to M via N . The problem of selecting of neighbors can be formally abstracted as follows.

Problem: Neighbor selection for minimal number of field divisions.

Input: A source node S , an NCG_S , and a variable a .

Question: Is there a selection of nodes from set (i) of NCG_S such that all nodes from set (ii) of NCG_S are connected to at least one of the selected nodes, and that the number of selected nodes is less than a .

This problem can be trivially reduced to the Sequence Covering problem [10] which is known as a NP-hard problem. Procedure *SelectNeighbors()* (Figure 5 b)) achieves selection close to the lower bound. We estimate the lower bound of this problem as the number of essential neighbors incremented by one if essential neighbors do not cover all nodes from set (ii) of NCG_S . *SelectNeighbors()* uses the fact that most of the nodes in a network can be covered by essential neighbors, or by non-essential neighbors that have the largest coverage count. The procedure first selects all essential neighbors. If there are nodes from set (ii) of NCG_S that are not covered, the procedure proceeds in a greedy fashion by selecting a non-essential neighbor that covers the largest number of remaining nodes from set (ii) of NCG_S . When all nodes are covered the procedure returns the selected values.

The *BuildBorder()* procedure is invoked when two regions are separated. Inputs to the procedure are two selected neighbors that are geographically adjacent, and nodes from set (ii) of NCG_S that are covered by two selected neighbors. The output of the procedure is a set of *Border Points* that describe the border. The first step of the procedure is to eliminate the nodes that have equal length shortest paths from the two selected neighbors. These nodes are considered as *don't-care* nodes from the perspective of the assignment to one of the two regions affiliated to the two selected neighbors. Geographical positions of *don't-care* nodes are, in most cases, between essential nodes, i.e. in the areas where the border are placed. When *don't-care* nodes are eliminated from consideration while building borders, more space is provided for placing borders. Consequently, borders have less border points and the compression ratio improves.

As a next step, *BuildBorder()* procedure sorts the nodes by their distance from source node S . It then determines

```

Calculate All-Pairs Shortest Paths
For each node i
  SelectNeighbors(i)
  For adjacent pairs of selected neighbors
    BuildBorder()
  End for
  BuildShortestPathTable(i)
End for

```

a) Procedure for building compressed shortest path routing table

```

Select Essential Neighbors  $EN_k$ 
Assign All Nodes  $I$  to  $EN_k$  such that
 $SPL(S, I) = SPL(EN_k, I) + 1$ 
While there is an unselected node
  Select non-Essential Neighbor  $NE_k$  with the largest
  number of shortest paths to unselected nodes
  Assign All Nodes  $I$  to  $NE_k$  such that
   $SPL(S, I) = SPL(NE_k, I) + 1$ 
End while

```

b) *SelectNeighbors()* procedure for source node S

```

CurrentSource = S
Repeat
  Eliminate all don't-care nodes of selected neighbors from the selection
  Sort selected nodes by distance from CurrentSource
  For all selected nodes
    Record minCW/maxCCW angle for two neighbors respectively
    If minCW < maxCCW
      Obstacle found
      CurrentSource = Node closer to CurrentSource corresponding to minCW or maxCCW
      Break
  End for
  Insert Border Point at closer node corresponding to minCW or maxCCW
Until no obstacles found
  Insert last Border Point

```

c) *BuildBorder()* procedure for two adjacent neighbors of source node S

Figure 5: Pseudo code for procedure of compression of shortest path routing tables. Pseudo codes for two backbone procedures of the main procedure are also shown.

initial $minCW$ and $maxCCW$ ¹ angles that correspond to the lines originating from source S and closest nodes affiliated with two neighbors (Figure 6). $MinCW$ is the angle of the line from source S and a selected node that is affiliated with the selected neighbor assumed to be in CCW direction from the other selected neighbor. Each of the selected and ordered nodes is considered. $MinCW$ and $maxCCW$ are updated during the procedure if necessary. If $minCW$ passes $maxCCW$ or vice versa (i.e. $minCW < maxCCW$), then the obstacle is detected. A node closer to the source node is selected as being one of the *Border Points*. The inserted *Border Point* is now considered as a new source node, selected nodes are sorted again by distance, and the algorithm is repeated starting from a direction setup by the previous inserted border of the region. The algorithm ends if there are no obstacles detected, and the node with the largest distance from the current source is chosen to be the last *Border Point*.

An example illustrating *BuildBorder()* procedure is illustrated in Figure 6. For source node S , two selected essential neighbors are nodes M and N where M is assumed to be in CCW direction from N . Nodes A, B, C, D , and E are essential nodes of S via M while nodes F and G are essential via N . Nodes H and I are *don't-care* nodes, and they are not considered during border building. Figure 6 a) depicts the initial $minCW$ and $maxCCW$ angles corresponding to A and F respectively. When D is reached, $minCW$ becomes smaller than $maxCCW$, and F is selected as a *Border Point*. The procedure is repeated but time obstacles are detected ($minCW > maxCCW$ for all selected nodes) and E is se-

lected as a *Border Point*. Note that if H and I were considered during building the border, additional points would be inserted. This would increase the size of the routing table. With our algorithm, *don't-care* nodes are automatically assigned to the region such that the size of the routing table is minimal.

3.2.1 Correctness and optimality of the algorithm

The following theorem validates the correctness of the representation of a network field divided into routing regions. Our algorithm relies on the fact that all routing regions are intersecting with the communication range of the source node.

THEOREM 1. *The circular sector CS , which is the result of the intersection of the area of communication range (circle area) of source node S and each of the regions derived from the field division routing table of S , has a non-zero area.*

Proof. With no loss of generality, let us assume that all selected neighboring nodes are essential neighbors to source node S . From Lemma 1 it follows that there exists at least one essential node E via neighbor N with $SPL(S, E) = 2$. By construction, our algorithm builds a border from the selected node closest to source node S . By Definition 1, initial values of $minCW$ and $maxCCW$ are such that $minCW > maxCCW$. Otherwise, E would not be essential, or it would be essential via another essential node of S . This holds for regions in both CW and CCW direction from region affiliated with N . Since there is at least one essential node E routed via N for each region, and since the initial $minCW$ and $maxCCW$ are not intersecting, it follows that CS does not have zero area. When among selected neighbors there

¹ $MinCW$ denote the "minimal clockwise" angle while $maxCCW$ denote the "maximal counterclockwise" angle.

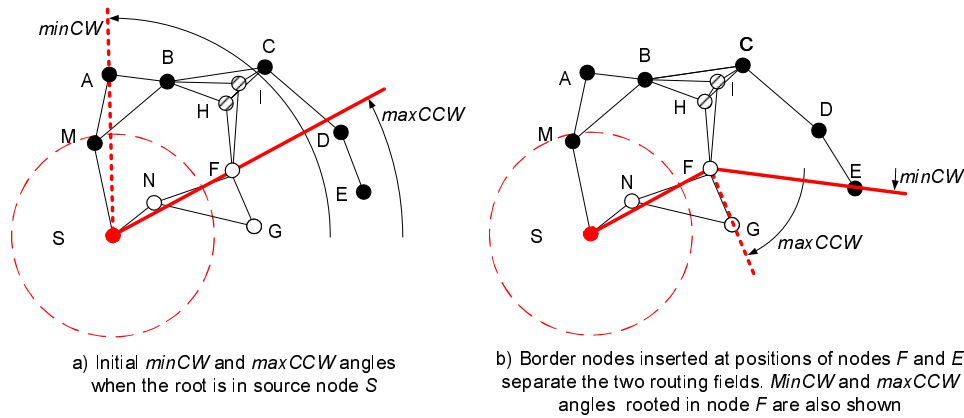


Figure 6: An example of $BuildBorder()$ procedure for source node S with resulting borders separating routing regions affiliated with essential neighbors M and N .

are non-essential neighbors, the proof can be trivially generalized by removing all neighbors of S that are not selected in which case all selected neighbors become essential. \square

THEOREM 2. $BuildBorder()$ procedure inserts the minimal number of **Border Points** which implies optimality of the shortest path routing table compression algorithm with respect to the selection of neighbors of source node S .

Proof. The optimality of the shortest path routing table compression algorithm is enabled with the removal of all *don't-care* nodes. Our algorithm inserts the number of *Border Points* equal to the number of crossings of $minCW$ and $maxCCW$ angles. This is equal to the minimum number of *Border Points*. Selection of node closer to the source node (see pseudo code in Figure 5 c)) guarantees that the border is placed on the tangent points of hulls that encompass regions without including any of the nodes from another regions. \square

Graphs that model ad hoc wireless network topologies are undirected. The graph density depends on the number of nodes and their communication range. We use Johnson's algorithm to calculate the shortest path. Thus, the runtime complexity of our algorithm for a graph $G = (V, E)$ is $O(V^2 \log V + VE)$ where V represents the number of nodes, and E represents the number of edges in the graph.

3.2.2 Deployment of routing tables

The deployment of the routing table is rather simple. One or more base stations send out the routing table to nodes within its communication range. The remaining routing tables are forwarded to nodes starting from nodes who already received tables from their neighbors. The deployment of tables is also inexpensive because routing tables contain minimal information which scales with the size of the network. When a node has the compressed shortest path routing table, it is necessary to discover to which region the destination belongs to inside of the routing table. For all borders we discover the position of the destination relative to the border. The relative position is a Boolean variable indicating either side of the border. Two adjacent borders in the routing table that have opposite relative positions indicate the region where the destination is. This procedure has complexity of $O(BP)$ where BP denotes the number of border

points. In the case of complex regions, the computation can become expensive for a node with limited computational resources. However, our compression scheme results in regions described with simple regions.

3.2.3 Update of field division shortest path routing table

It is a common case that node locations in a wireless ad hoc network can change, inducing changes in network topology. Such changes affect the way nodes forward their messages because of the change in the routing tables. Field division routing tables are updated locally, by nodes, without the assistance of base stations.

Source node S considers the modification of its field division routing table in one of the three following cases: (i) one of its neighboring nodes is no longer reachable with a single hop, (ii) a new node enters the communication range of S , or (iii) a neighboring node changes its routing table. In cases (i) and (iii) the routing table changes only if the node that left the communication range or changed its routing table is an affiliated neighbor with a routing region of S . Based on Theorem ??, in case (iii) the routing table is changed if and only if the new node introduces an essential node routed via the new node.

When node P leaves the communication range (case (i)), the trivial case of update of the routing table is when one of neighbors of S completely takes over the role of P . That node can be either an already selected or unselected neighbor of S . If the neighbor taking over the routing region of P is an already selected neighbor, one of the borders in the routing table is erased. If it is necessary to split the routing field of P between two selected neighbors (Figure 7), S pools nodes within the region for the shortest path distance via two selected neighbors. Then the border is proactively built in an iterative fashion. Alternatively, the border can be built reactively, as the request for forwarding in the particular routing region are received.

When a new node P enters the communication range of S (case (ii)), and a new routing region affiliated with node P needs to be introduced (Figure 8), then a new border is built. The new border will lie in the unresolved region, which is an overlap region between routing regions of P and N where N is a neighbor of S that is already in the routing

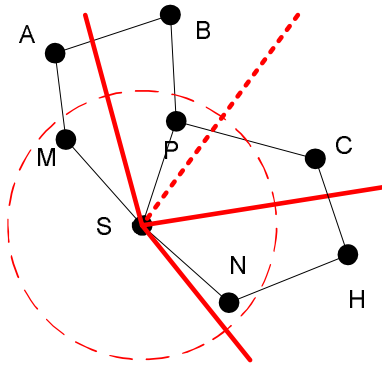


Figure 7: An example of update of routing table of source node S when node P leaves communication range of S , and the routing range is split.

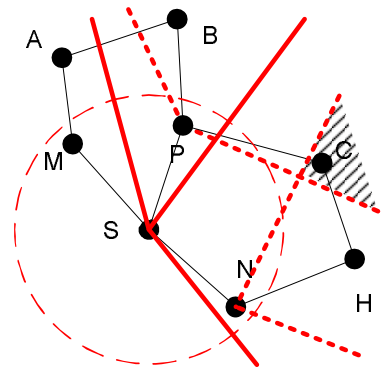


Figure 8: An example of update of routing table of source node S when node P enters communication range of S , introducing the new routing region.

table. Overlapping routing regions cover only the nodes that encompass neighbors of nodes P and N that are essential from their perspective. This observation is a consequence of Lemmas 1 and 2. Such unresolved routing region is illustrated in Figure 8 as the shaded area. The procedure of border building proceeds in the standard way while considering only unresolved routing regions. If an unresolved routing region fully covers the routing region of either node, new node P takes over that routing field completely, and a new border is not built. Nodes in unresolved region are pooled in order to determine their shortest path distance from nodes P and N . While pooling nodes, the unresolved region is updated and it quickly converges to only nodes that need to be considered for building the border inside the initial unresolved region. This property enables efficient scaling of the algorithm for building borders. Case (iii) can be reduced to case (ii) if the neighbor that has changed its routing table is considered as a new node in the communication range of S .

The complexity of the algorithm for updating depends on the level of mobility of nodes, movement velocity, and network density. During the update of borders, a portion of network nodes is pooled for the new shortest path routs. This indicates that the complexity of the algorithm is sub-linear in terms of the total number of nodes in the network. A more precise complexity analysis requires detailed simula-

tions over a range of densities, node velocities, and the level of mobility of nodes.

3.3 Energy Balanced Routing

In field division shortest path routing, a node or a point within a sensor network is deterministically located within a single routing region from the perspective of source node S . This is the consequence of non-overlapping routing regions. There are a number of nodes referred to as *don't-care* nodes whose shortest path route can go via more than a single neighbor of source node S . Field division shortest path routing assigns *don't-care* nodes automatically to one of the routing regions such that the routing table is compressed optimally.

We modify the initial requirements for the fully compressed routing table such that the field division shortest path routing table includes information about which neighbors can be reached via multiple nodes. Simultaneously, the routing table must maintain the requirement for shortest path routing. These requirements are satisfied if routing regions overlap. The routing table is modified compared to the structure presented in Figure 3. In the modified routing table, each neighbor from the *List of Neighbors* is associated with two borders each containing a number of *Border Points*. This increases the size of the border compared to the minimal case to twice its size because no borders are shared among selected nodes from the *List of Neighbors*. Since previously neglected *don't-care* nodes are included during the procedure of building borders, the number of *Border Points* also increases. Consequently, in cases with denser networks and network with a large number of nodes, the routing table size is less scalable (Table 2).

In energy balanced routing, each node makes a localized and dynamic decision via which neighbor it should forward its message. Forwarding a message to essential nodes is still restricted to forwarding via a single node. However, because of the fact that there can exist multiple paths to essential nodes (e.g. in Figure 4 $N - A - C$ and $N - B - C$ are both shortest paths from source node S to node C) the load balancing can be achieved throughout the network while being based only on localized decisions. We are currently investigating an additional step or constraint relaxation where a shortest path can be extended by n hops. This further increases the number of *don't-care* nodes that are covered with multiple routing regions. Consequently, the flexibility of balanced routing is higher while simultaneously the routing table size is reduced because of the more flexible areas where it can be placed. Note that algorithms for the reduction of energy consumption in wireless ad hoc networks by switching off transmitters of certain nodes such as GAF [20] can be applied as a higher order application layer. Similarly, a variable communication ranges of nodes can be considered as an option to balancing out their communication load.

4. TRAJECTORY COMPRESSION

Another example where model-based compression can be applied is to the compression of trajectories, which is closely related to tracking objects. Tracking objects is a common task in sensor networks [8, 21]. Often the goal of tracking is to predict the position of the observed object. Such a goal can be achieved with predictors derived from general purpose compressors [1]. Object tracking is typically associated with a large number of objects so the resulting data

sets are large. We present an algorithm for trajectory compression with the main goal to reduce the size of the stored trajectories. As before, the compression engine can be used to prediction a trajectory. A step further is to characterize the object that is tracked, based only on the observation of the compressed trajectory. Our compression algorithm improves compression rates that are correlated with data entropy. Entropy is the measure of the amount of information in a data set [19]. Because of the improved compression ratio, the amount of information about the object is better defined. Consequently, we can characterize tracked objects. For example, we can determine if the object tracked is of artificial intelligence or piloted with an automated pilot when trajectories are compressed and compared with a database of trajectories. The difference in compression ratios of the trajectories can reveal other useful information about the tracked object and its possible behavior.

Compressing the trajectory with one of standard lossless compression engines (e.g. [5, 22]) yields compression rates comparable or better than when compressing ASCII encoded textual files. However, the quality of compression rates in this case can be misleading. Trajectories are usually represented as a set of numbers encoded in ASCII format which uses 8 bits for each digit while the total number of digits is equal to 10. This redundancy in encoding can be removed by general purpose lossless compression engines can. At the same time, they are doing a poor job on capturing actual correlations embedded inside of trajectories, i.e. their entropy. This is essential for other uses of compressed files other than for the minimization of file size.

We have developed an algorithm to capture correlations within trajectory data. Our algorithm improves compression rates compared to other compression engines. As the first step (Section 4.1), we have modeled the trajectory data representation to capture idiosyncracies of trajectories, while significantly reducing the encoding overhead. Furthermore, this type of encoding enables better modeling within our compression engine. The second step (Section 4.2) of our compression algorithm uses three weighted Markov models to compress the encoded data.

4.1 Trajectory Data Transformation

In most applications, it is assumed that the trajectory is given by a set of absolute points that corresponds to a grid overlaying the total coverage of the sensor network area. The grid is usually rectangular and often square, and each cell in the grid is assigned a label in the form of numbers in ASCII representation. The set of such points in a certain order represent a trajectory. The starting point of our algorithm is to transform the model of trajectory information such that it represents the directions and their changes, rather than their absolute positions. Relative positions of points in the trajectory reveal correlations in the behavior of the observed object. Absolute positions can be useful for example in characterization of objects if they are closely related to certain locations of interest within the coverage area of the network. This information on the representation of relative positions are not lost however. In fact, the relative position of location of interest in the network coverage area affect in the same way the data in the relative representation of the trajectory.

Another effect of trajectory data transformation is the removal of redundant encoding. Relative positions can be encoded efficiently with a few bits. This effect is similar to

the effect when instead of 8 bits for each ASCII character used, we use a minimal of 4 bits per character since in the description of absolute positions we need only 10 digits and one separator. Other redundant information in the absolute position trajectory representation are the information about the grid. This information should be implicit because the structure of the grid is known.

In our trajectory data transformation algorithm, we have assumed that the difference in locations of two adjacent positions can be only a single cell in the grid, where two cells positioned diagonally have the distance of two cells. In this kind of encoding of trajectories, there are only 4 different directions where one point can be located compared to another adjacent one: **up**, **down**, **left**, and **right**. These 4 directions can be encoded minimally with 2 bits. Diagonal movements are actually encoded with 4 bits.

4.2 Trajectory Data Compression

Trajectory data transformations described in the previous section enabled trajectory data compression to take place in such a way that it can reach trajectories' entropy by transforming a set of locations into a set of directions. A rule of thumb in data compression is that better prediction of upcoming data in the input stream, leads to better compression rates. The prediction of the next location in trajectory data compression corresponds to the prediction of the next symbol in data stream in general purpose compression. Compressors that yield the best compression rates are known to be compressors based on Prediction by Partial Matching (PPM) [5] which use the history of appearance of symbols in particular contexts to predict upcoming symbols. Arithmetic coding does the encoding part of PPM.

We have exploited the idea of using symbols in previously seen contexts to predict upcoming symbols for our compressor of trajectory data. However, the context prediction cannot be directly applied in this problem. In text compression for example, a particular context of symbols is partially enforced by certain grammar rules of a language. A particular order of directions does not give a high likelihood to the appearance of exactly the same order of directions. We have identified three appropriate contexts for describing and predicting trajectory behavior as local, regional and global history of trajectory. We collect the statistics of the trajectory behavior in the last 16 moves (local history), last 256 moves (regional history), and all the moves seen so far. We weight them and then combine them in order to predict the next move. Finally, this prediction is passed to a standard arithmetic coder. The length of local and regional buffers depends on the granularity of the grid and can be the algorithm parameters. Weights are determined empirically, and also can be the algorithm parameters.

5. EXPERIMENTAL RESULTS

5.1 Field Division Routing Results

In this section, we present the experimental results of the field division routing table compression. We have tested the effectiveness of our algorithm on a set of networks whose nodes are randomly positioned with uniform probability distribution. We present results for networks of size 100, 200 and 500 nodes with different communication ranges. For each network setting, the communication range is the same for all nodes. The communication range is given in units

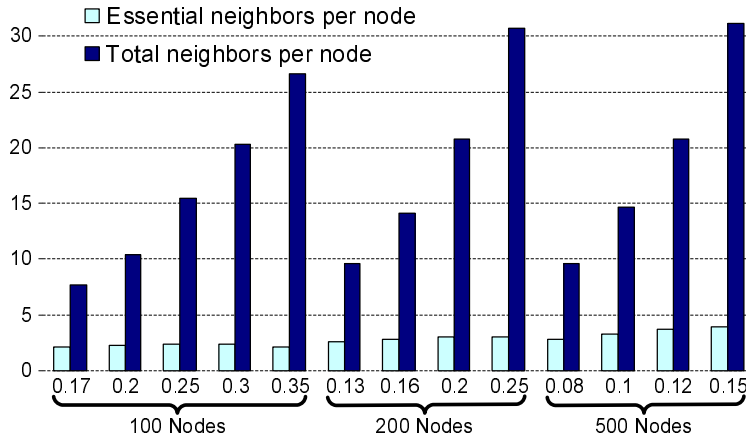


Figure 9: Network statistics. The graph displays the average number of essential and total neighbors per each node for 13 networks with 100, 200 and 500 nodes. Network also differ in their respective communication ranges which is displayed directly below bars for each network. The communication range is normalized over the length of x dimension of the network field (considered to be 1.0).

relative to the length of the x -axis. We assume a planar, square network field of unit size. Figure 9 depicts the statistics for 13 network settings where 5 of 100 nodes, 4 of 200 nodes, and the last 4 have 500 nodes. Network density is affected by different communication ranges of nodes, which range from 0.08 for 500 nodes to 0.35 for 100 nodes networks. All presented results are averaged over 20 different networks for each setting, and then averaged per single node. The average number of total neighbors per node ranges from 7.7 for networks of 100 nodes and communication range of 0.17, to 31.1 for networks with 500 nodes and communication range of 0.15. A typical standard deviation of the total number of nodes for a network with 100 nodes and communication range of 0.17 is 2.58 for 7.72 average total number of neighbors. The standard deviation decreases relative to the number of nodes for larger and denser networks. The minimum communication range is chosen such that the network is connected, i.e. any node can communicate with any other node in the network via a number of communication hops. The communication range is increased to the point when the density becomes high.

Figure 9 displays network statistics for the number of essential neighbors per each node. An interesting result shown in Figure 9 is that the rate of increase of the number of essential neighbors significantly lower than the rate of increase of the number of nodes. For example, for networks with 100 nodes and communication range of 0.17, the average number of essential neighbors is 2.19 per node. For networks with 100 nodes and communication range of 0.35, the average number of essential neighbors is 2.15 per node. At the same time, the average number of total neighbors per node increases from 7.7 to 26.6 per node, or 3.5 times. Similar behavior is recorded for networks with similar density (approximately the same number of total neighbors per node) but with different number of nodes. When these networks have on the average 10 neighbors per node, the average number of essential neighbors is 2.30, 2.58, and 2.83 for networks with 100, 200, and 500 nodes respectively. In other words, the number of essential neighbors increases 23% while the number of nodes increases 400%.

Compression ratios of our algorithm improve if procedure *SelectNeighbors()* returns as few as possible neighbors. Table 1 illustrates the difference between the lower bound and the result of our algorithm. A possible lower bound would be the number of essential neighbors. However, when essential neighbors do not cover all nodes, at least one additional neighbor must be inserted. We consider that number as a better lower bound for the estimate. From the results in Table 1, we can see that our algorithm is in most cases within one third off the lower bound estimate. Only in one case our algorithm produces close to two thirds over the lower bound. Note that the accuracy of the lower bound estimate decreases as the network density increases due to the fact that the number of essential nodes decreases relative to the number of non-essential nodes. This is illustrated by the percentage of essential nodes in the network in the third row of Table 1.

Finally, we compare the estimated compression ratio obtained in our compression scheme as the number of bytes necessary to represent the field division shortest path routing tables versus a representation of shortest path routing table that has the assignment of every node in the network to one of its neighbors. This comparison is fair because these two representations contain the same information without redundancies. By definition, the compression mechanism removes redundancies in the representation of data which, at first glance, contradicts the previous statement. However, our compression scheme removes redundancies that are in the modeling of the network representation data rather than the encoding of the table. We assume that our routing table represent each routing region with 5 bytes: 4 of them to represent the location of the neighbor affiliated with the region (2 bytes for each coordinate), and 1 additional byte to describe the region (open or closed region) and adjacency relations of selected neighbors. Each *Border Point* location is represented with 4 bytes. A full routing table uses 4 bytes to represent neighbors and 4 additional bytes for the location of each node that is not a neighbor. We neglect other fixed costs that might be necessary for proper description of both routing tables. We have assumed the same number of neighbors for the description of full routing table as in the

Nodes	100					200				500			
Range	0.17	0.20	0.25	0.30	0.35	0.13	0.16	0.20	0.25	0.08	0.10	0.12	0.15
Ess. nodes [%]	65.9	58.1	44.9	40.4	42.0	58.1	43.3	32.4	27.1	56.1	38.4	27.5	15.4
Ess. neigh.	2.19	2.30	2.39	2.37	2.15	2.58	2.85	3.02	3.06	2.83	3.31	3.67	3.98
Lower bound	2.75	2.94	3.16	3.16	2.94	3.19	3.59	3.82	3.91	3.49	4.06	4.47	4.83
Sel. neigh.	2.93	3.26	3.70	3.86	3.74	4.26	4.59	5.07	6.46	4.37	4.68	5.24	5.87
Diff. [%]	6.7	11.0	17.3	22.1	27.4	33.4	27.6	32.8	65.4	25.2	15.3	17.3	21.4

Table 1: Average difference of the number of selected neighbors which is equivalent to the number of routing regions in our routing table from the estimated lower bound.

Nodes	100					200				500			
Range	0.17	0.20	0.25	0.30	0.35	0.13	0.16	0.20	0.25	0.08	0.10	0.12	0.15
FDRT [B]	27	30	34	35	34	40	42	46	59	44	44	49	54
Full RT [B]	377	368	350	331	305	775	758	734	700	1976	1956	1934	1895
CR [%]	7.2	8.1	9.6	10.5	11.1	5.1	5.5	6.3	8.3	2.1	2.2	2.5	2.8
Rel.FDRT [B]	53	69	81	89	87	86	112	137	151	117	168	207	246
Rel.CR [%]	14.0	18.7	23.0	26.9	28.4	11.0	14.8	18.6	21.5	5.9	8.6	10.7	13.0

Table 2: Compression ratio (CR) of field division routing table (FDRT) vs. full routing table (Full RT) whose sizes are given in bytes are illustrated. Compression ratio when constraints are relaxed (Rel. CR) vs. full routing table is also presented.

full division routing table. We average the result for each network setting over 20 different and connected networks.

The row labelled as CR row in Table 2 illustrates the obtained compression ratios. The rates improve with the increase in number of nodes in the network, which indicates that our algorithm is scalable. The largest number of bytes in a full division routing table is 59 for a given set of networks, while the largest full routing table would have almost 2000 bytes in the worst case.

The row labelled Rel.CR in Table 2 illustrates the compression ratio when we relax constraints as described in Section 3.3. Routing tables are multiple times smaller than the original routing table size. This holds even for networks with a very large number of nodes, where the largest routing table size reaches 246 bytes. Note that in many cases, load balancing can be efficiently applied because of large number of nodes that can be routed via multiple neighbors (in most of the cases the ratio is over 50% - see Table 1). Our preliminary experiments with static path reassignment estimate 10% increase in time before the first failure of node.

5.2 Trajectory Compression Results

In this section, we present results of our Trajectory Data Compression algorithm. We show the results of compression of seven different trajectories. Three of them are generated by tracking data of objects in a sensor network when objects are traversing the network on its minimal exposure path. The remaining four are generated as a random walk inside of the same grid. The results are presented in Table 3. We compare our approach with an LZ based compression algorithm [22]. The third row contains compression ratios when we compress files with the LZ compressor, and the fourth row when compressed with our algorithm. Our algorithm outperforms LZ by a huge margin because of the redundancies in the representation are removed from files. Our scheme is able to exploit all correlations of data in trajectories. We have compressed the transformed files with

both the LZ compressor and the second step of our algorithm. These results are presented in the fifth and sixth rows respectively. While both algorithms do not compress random data (the size increases) as expected, our algorithm outperforms LZ because it captures better idiosyncrasies of trajectory data on the average. More importantly, our algorithm classifies data better than LZ. This is illustrated by the standard deviation of compressed file size shown in the last column of the fifth and sixth row. The standard deviation is normalized on the initial file size. Since we have two distinct sources of information (minimal exposure and random walk), it is expected that we can compress files with almost no deviation in compression ratios. With that respect, our algorithm outperforms LZ by an order of magnitude.

6. CONCLUSION

We have presented two techniques for modeling data for compression in wireless ad hoc networks. The first technique enables efficient and scalable shortest path routing. The scalability of the routing table size is a consequence of properties of ad hoc networks where the number of important regions from the perspective of the shortest path routing does remain nearly constant while the size and the density of the network increases multiple times. We have efficiently exploited such properties to build compact and simple routing tables which are 10 to 45 times smaller for the selected range of networks than the full and uncompressed shortest path routing tables. We have also investigated possibilities for balancing the load of the routed traffic for better utilization of scarce energy resources in sensor networks. The second technique models trajectory data to achieve superior compression rates. The compression is done in two steps where the first step removes the encoding redundancies of standard trajectory representation. The second step is a compression algorithm that efficiently exploits the correlations of trajectory data. Our compression scheme consistently outperforms LZ compression scheme in

Trajectory type	Minimal Exposure			Random Walk				Normalized St. Dev.
	MinEx1	MinEx2	MinEx3	RW1	RW2	RW3	RW4	
File Size [KB]	38	12	18	96	16	65	429	
LZ CR [%]	26.0	29.6	31.7	15.9	16.7	16.1	15.6	
TDC CR [%]	1.3	1.2	1.2	3.1	3.2	3.1	3.0	
LZ CR over Transf. [%]	42.5	66.0	60.0	103.9	123.6	105.9	100.9	22.5
TDC CR over Transf. [%]	42.6	39.4	41.1	101.3	102.2	101.4	101.0	2.1

Table 3: Performance comparison of Trajectory Data Compression (TDC) algorithm vs. LZ based algorithm. Compression ratio (CR) of LZ and TDR are ratios of compressed and original file sizes. Last two rows represent CRs when LZ and TDR are run on file transformed to its relative position.

terms of compression rates and consistency in compression rates from data coming from the same type of source.

7. ACKNOWLEDGEMENTS

We thank professor Margaret Martonosi and the anonymous reviewers for discussions that improved the content of the manuscript. This material is based upon work supported in part by the National Science Foundation under Grant No. ANI-0085773 and NSF CENS Grant.

8. REFERENCES

- [1] A. Bhattacharya and S. K. Das. Lezi-update: An information-theoretic approach to track mobile users in PCS networks. *Mobile Computing and Networking*, pages 1–12, 1999.
- [2] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. *INFOCOM The Conference on Computer Communications*, 1:22–31, 2000.
- [3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–94, 2002.
- [4] P. B. Chu, N. R. Lo, E. C. Berg, and K. S. Pister. Optical communication link using micro corner cube reflectors. *Micro Electro-Mechanical Systems*, pages 350–5, 1997.
- [5] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.
- [7] J. Deng and Z. J. Haas. Dual busy tone multiple access (DBTMA): A new medium access control for packet radio networks. *IEEE International Conference on Universal Personal Communications*, pages 973–7, Oct. 1998.
- [8] Z. Deng and W. Zhang. Localization and dynamic tracking using wireless-networked sensors and multi-agent technology: First steps. *IEICE Transactions on Fundamentals of Electronics Communications & Computer Sciences*, E85-A(11):2386–95, Sept. 2002.
- [9] J. J. Garcia-Lunes-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking (TON)*, 1(1):130–41, 1993.
- [10] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, 1979.
- [11] Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. *IEEE International Conference on Universal Personal Communications Record*, 2:562–6, 1997.
- [12] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. *IEEE International Multi Topic Conference*, pages 62–8, 2001.
- [13] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*, chapter 5, pages 153–81. Kluwer Academic Publishers, 1996.
- [14] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. *Mobile Computing and Networking*, pages 243–54, 2000.
- [15] S. Murthy and J. J. Garcia-Luna-Aceves. A routing protocol for packet radio networks. *Mobile Computing and Networking*, pages 86–95, 1995.
- [16] V. D. Parkand and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *INFOCOM The Conference on Computer Communications*, 3:1405–13, 1997.
- [17] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, pages 234–44, 1994.
- [18] C. Perkins and E. M. Royer. Ad hoc on demand distance vector (AODV) routing. *IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb. 1999.
- [19] I. Witten, A. Moffat, and T. Bell. *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann, San Francisco, Ca., 1999.
- [20] Y. Xu, J. S. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. *Mobile Computing and Networking*, pages 70–84, 2001.
- [21] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, Mar. 2002.
- [22] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions Information Theory*, IT-23(3):337–43, 1977.