Ad-Hoc Routing Using Virtual Coordinates Based on Rooted Trees

Yosi Ben-Asher, Moran Feldman and Sharoni Feldman

Computer Science Department, Haifa University, Israel
{yosi,sharoni}@cs.haifa.ac.il

Abstract

An ad hoc routing algorithm must be able to locate and maintain a path from a source to a destination node while overcoming sudden changes in the network topology and explore alternative paths in case of path breaks. One "classic" type of solution is to constantly maintain an approximate routing table at each node and use it to dynamically search for a routing path to the destination. Another, more efficient type of algorithms uses geographical/geometrical coordinates at each node (obtained by GPS devices) to forward packets and explore alternative paths to the destinations. Recently it was shown that the exact GPS coordinates can be replaced by "virtual" grid like coordinates computed based on the number of links/hops between the nodes. Computing these grid-like coordinates is a costly process eliminating the advantages of the geographical routing over the more classic types of algorithms. We propose a different type of virtual coordinates based on dynamic embedding of the ad-hoc connectivity graph by a minimal set of rooted trees. We give a novel ad hoc routing algorithm called MRA (Metrical Routing Algorithm) that uses the tree like coordinates to efficiently find minimal cover of the nodes by rooted trees and based on the tree coordinates forward packets to their destinations. Similar to geographical routing our algorithm allows oblivious transfer of packets to any destination avoiding the use of routing tables.

The proposed method is compared to the well known AODV (Ad Hoc on Demand Distance Vector Routing) obtaining significant improvements in terms of number of messages, number of concurrent sessions, duration of the sessions, nodes' densities, queue sizes and resilience to sudden movements of the nodes. We have built a sophisticated simulator that has been carefully designed for a fair comparison between the two algorithms

1. Introduction

Ad Hoc networks allow communication between a dynamic set of mobile wireless users without using a fixed set of base stations. Each user of an Ad Hoc network also acts as a router allowing other users to communicate through its mobile communication device. The communication range of each device is limited thus at any given time a user can exchange packets only with other devices in its receiving/transmitting range. The set of users is highly dynamic: new users join in while other users may quit or move out of receiving range. In addition each node can arbitrarily move and possibly cause loss of communication with some nodes while creating new connections with other nodes. The basic communication structure is therefore of a dynamic graph where new edges (communication links with neighbors) and nodes are fused while other nodes and edges get arbitrarily deleted. This extremely difficult setting demands novel communication protocols that can maximize the amount and length of “sessions” between the users.

The main or the "classic" types of ad-hoc routing algorithms (see [6]) maintain routing tables at each node containing the path (where to forward packets) for every possible session. Each change in the network (e.g., a communication link with a neighbor has been disconnected) must be forward to all other nodes in the network so that they can update their routing tables. Another possibility of the classic types is to dynamically search for a given destination. Typically such a search may require flooding the network by “search messages” until the destination is reached and a routing path has been located.

The AODV algorithm [3] combines on demand search with temporal recent destination routing information and is considered to be one of the most efficient algorithms for ad-hoc routing. In AODV a search for a route to a given destination is basically

performed by flooding. In AODV each node maintains a list of all its recent routings, thus the flooding can be reduced when it reaches an intermediate node that has the destination in its recent routing list. It becomes obvious that the AODV and all the other "classic" algorithms may use quadratic number of messages (in the number of nodes) to perform a search for a route or to update a routing table.

Another type of ad-hoc algorithm uses the GPS coordinates of each node to forward packets to the neighboring node that has the shortest Euclidian distance to the destination. The routing is oblivious and thus potentially more efficient than the one used in the AODV types of algorithms. Using GPS can be regarded as a way of assigning grid like coordinates (in the real plane) to every node and using the grid metric to select the shortest routing paths.

Compared to AODV type of algorithms in Geographical routing there is a need to locate the geographical coordinates of every destination before a packet can be sent. One solution was proposed by the GLS algorithm [5] wherein each node *v* periodically broadcasts its coordinates to a small well dispersed subset of nodes that have the nearest id-number to *v*. This subset of nodes forms a search tree allowing relatively efficient search for nodes' coordinates. Methods for locating coordinates based on nodes' ids also include the use of distributed hashing (DHT) see[9]. Another interesting algorithm is DREAM[4] where coordinate updates to far away nodes are less frequent than updates to nearer nodes. The algorithm that is proposed here also coordinateses this problem but uses a different type of solution to it. One problem of forwarding packets to the geographically nearest neighbor is that the packet may reach a dead-end where all the neighbors are far apart from the destination than the current node. In this case the packet must move "backwards" along the current "face" searching for a new node out of which the greedy forwarding can continue (e.g., the GOAFR+ [13] algorithm combining greedy routing and face routing). A Pure face routing algorithms have also been proposed and analyzed, e.g., Compass routing [2], GFG[8],GPSR[7] and AFR[4].

Figure 1 left side demonstrates a bad case of face-routing where a packet from node S should reach node T. The gray nodes indicate real nodes and the transmission distance is such that each node is connected only to its left, right, up down most nearest grey neighbors. Following the short distance to T, the packet first moves from S to L but then must move along the face back to Q. At this point the packet takes the longer way to T and moves down since the lower neighbor of Q is closer to T than Q's upper neighbor.

Recently, Rao et. Al. [12] proposed to use virtual coordinates that approximate geographical coordinates without using GPS. The basic idea is to embed the nodes in a grid wherein Euclidian distances in the plane are replaced by connectivity distances in the dynamic communication graph. The algorithm has three stages: 1) The nodes on the perimeter are identified, 2) The virtual coordinates of the perimeter are computed, and 3) Based on the perimeter, virtual coordinates of internal nodes are computed. Virtual coordinates are basically computed using a relaxation method wherein each node updates its coordinate based on the coordinates of its neighbors. Other works on virtual coordinates assume that a subset of the nodes (anchors) are equipped with GPS and thus based on the anchors coordinates other nodes can learn their relative approximated coordinates. Note that the bad case of Figure 1 is avoided when using a spanning tree rather than the grid metric. Intuitively, the term metric can be defined as follows. Let G be a communication graph such that there is an edge between every two nodes that are in communication range. For ad hoc routing, A metric D can be intuitively defined as a sub graph of G "more simple" with coordinates such that the shortest path between every two nodes U,V can be computed based on their coordinates d(U,V)= f(U.coor,V.coor).

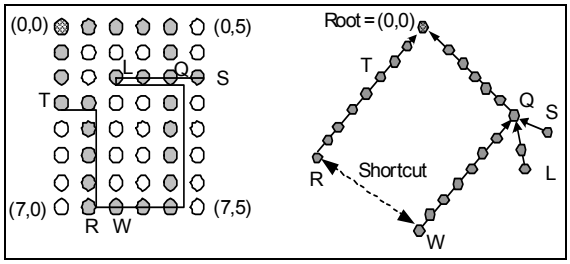

Figure 1: Face routing & tree routing

Actually the proposed metric uses spanning trees with "horizontal" edges (called shortcuts) connecting every node in the chosen rooted tree to all other nodes that are in transmission range. In routing from s to t, a shortcut edge <u,v> is used only if the distance from v to t on the rooted tree is shorter than the distance on the rooted tree from u's father and sons to t. Figure 2 illustrates this metric of rooted trees with shortcuts (denoted by double arrows).

Note that the short distance (based on the proposed metric) from s to t is obtained by going up to node h then through the shortcut to node j and finally to t. A shortcut will be created automatically when it can reduce the path length and there is no management of shortcuts in tables.

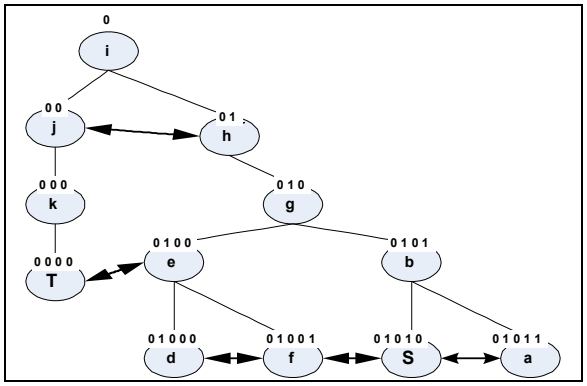

Figure 2: Routed trees with shortcuts

Note that potentially the packet could use a shorter path S➔f➔e➔T, however the distance in the tree from b to T is shorter than the tree distance from f to T. Note that the tree distance between any two nodes can be directly computed based on their IDs (their virtual coordinates in the tree), e.g., the distance between S=01010 and d=01000 is 4 as we need to go up 2 times and reach the common node/prefix g=010 and then down 2 times. There are of course bad cases where this metric (rooted trees with shortcuts) will fail and two relatively close nodes will be far apart (as is depicted in figure 2), however we believe that practically for most pairs of nodes the distance assigned by this metric will be close to the real one.

The choice of using the tree with shortcuts metric rather than the grid metric is motivated by the fact that it is better suited to handle the dynamic case of ad hoc networks. In particular note that adding a new node to an existing rooted tree requires only changing the virtual coordinates (IDs) of that node alone while adding a new node to an existing grid may require changing the virtual coordinates of most of the grid's nodes. In the case that a node U gets deleted, we might be forced to update the virtual coordinates of every node in the sub tree of U, however, usually the number of updates caused by deletion in the proposed metric is significantly less than that needed by the grid-like metric. In particular adding and deleting several nodes at the same time is more complex in grids than in trees as the new coordinates of most nodes in the grid are affected by all the changes while in trees the changes in two disjoint sub trees do not affect one another. Figure 3 presents a communication graph organized as a grid metric and a tree metric. Adding node A after the addition of node B in the grid metric requires a broad update of coordinates. The addition of nodes in the tree metric is totally independent.

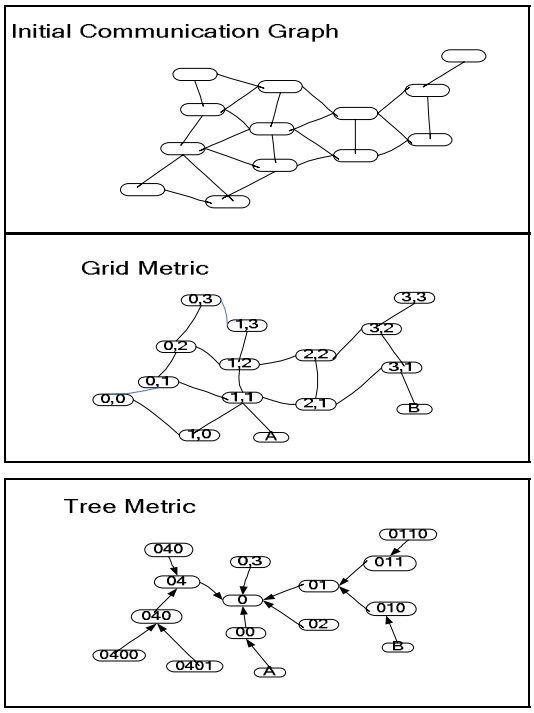

Figure 3: Spannig tree vs. grid metric

The choice of rooted trees metric is also motivated by the theoretical results of the "metric labeling" problem [15]. In this problem, the goal is to approximate distances of nodes in the plane via simple graphs with virtual coordinates. A known result for metric labeling with trees [14] is to select the "center" point in the plane as the root of a tree and keep adding new leaves by selecting points that are the closest to a given leaf (also known as Dijkstra's trees). The average distance distortion factor (ADDF) using Dijkstra's trees is less equal 2. Another result [16] shows that if we select a spanning tree at random out of a special distribution then the ADDF is less equal square log N, where N is the number of points in the plane. Better approximations to the static metric labeling problem can have been obtained, e.g., the use of spanner graphs or using integer programming [15], however, as explained such constructions are not feasible for ad hoc routing. In any case, the metric of rooted tree with shortcuts has not been proposed in this context.

Recently, [17] proposed an ad-hoc routing technique that is also based on routing via rooted trees where each node gets a virtual address in a rooted tree. This algorithm is not based on metrical routing and actually does not use virtual coordinates in the sense we have defined, i.e., embeds the connectivity/radius graph in a rooted tree such that the distance distortion is minimized.

The focus of this work is to obtain a pragmatic general proof for using the tree + shortcuts metric for ad hoc routing, hence theoretical issues such as formal proofs for the effectiveness of the proposed metric compare to the grid metric are not considered. We have developed a complex simulator for testing the proposed algorithm and comparing its performances with those of the AODV. The complexity of Rao et. Al. [12] is too high for a practical implementation leaving the AODV as the only competitor for the proposed method. We remark that every effort was made to design the simulator such that the comparison will be fair. The design of a suitable ad hoc simulator required careful consideration of many factors. The current simulator deals with the new Metrical Routing Algorithm (MRA) and AODV. It presents a set of enhanced visualization, real-time tracing and online control features. There are only very few simulators that can simulate ad hoc networks (e.g.,PARSEC[10], NS-[11]), however due to the special features of the proposed algorithm we have decided to develop a special ad hoc simulator. The results obtained by the simulator indicate that using tree based metrics for ad hoc routing can improve both the number of messages and the duration of the sessions between ad hoc nodes.

2. MRA Algorithm

In this section we describe the MRA algorithm, by presenting the notations, the data used to control the connections between the nodes and the session mechanism.

2.1 Main idea dynamic fusion of trees using shortcuts

The MRA algorithm organizes the nodes in the field in rooted trees. Only nodes that belong to the same tree can create sessions among themselves. To ensure the maximal connectivity, all nodes will try to organize themselves in a single tree. Every node in the field has a unique node-id (phone number or IP address) and virtual coordinates that may change depending on the changes in the tree structure. Every tree is identified with a "tree name" which is the id of the root node. Nodes send periodically beacons (hello messages). Every node that receives a beacon checks whether the node that sent the beacon belongs to a different tree. If the nodes belong to different trees, they will initiate a fuse process that will fuse the separate trees into a single tree. The fusion protocol should satisfy the follow properties:

1. The protocol should not cause active sessions to break
2. Eventually (assuming no dynamic changes occur) all trees with nodes within transmission area must fuse into a single tree.
3. When two trees are being fused most updates should be made to the nodes of the smaller tree (in the number of nodes).
4. The protocol should maximize the number of nodes that migrate from one tree to the other in every step (yielding a parallel fuse).
5. Nodes constantly attempt to shorten their distance to the root of the tree by fusion to higher level nodes.
6. Initially every node forms a separate tree of size 1.
7. The protocol is fully distributive with no "central" bottlenecks, namely it is defined at the level of pairs of nodes.

Every node in the tree can initiate a fusion process to a neighboring tree regardless of the node position in the tree. The fusion node gets a new coordinates in its new tree according to the node new position. Naturally, when a node migrates from one tree to a new tree, it may carry its neighboring nodes to follow him. Figure 4 presents two stages of the trees fusion protocol: the initial state and final organization to trees (assuming no dynamic changes occur). Note that the two separate trees in Figure 4-B cannot fuse because there are no two nodes between the trees within a transmission range that will start a fusion process.

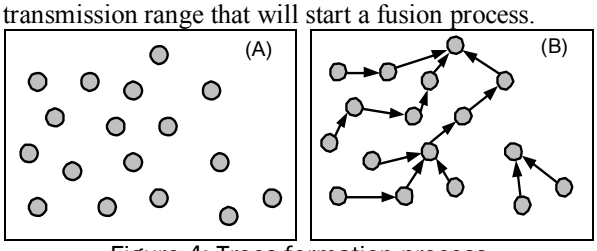

Figure 4: Trees formation process

The fusion process of two trees is a parallel process where at any given stage more than one node of the smaller tree joins the larger tree as is depicted in Figure 5.

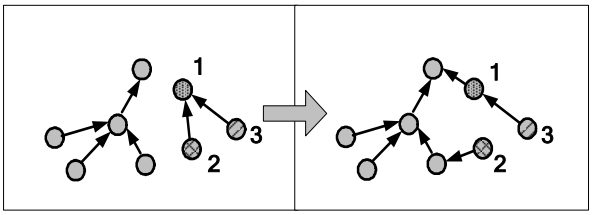

Figure 5: Fusion of trees

2.2 Notations

An Ad Hoc network is built of moving nodes. All nodes are identical and run the same software. Every node sends periodically a "Hello" message that is received by all nodes within the transmission radius.

Every node receives Hello messages from nodes within their transmission range. A "Source node" is a node that initiates a session that connects the "Source node" to the "Target node". The following notations are used:

Node Identification (*ID*) is a unique identifier of each node in the system. When a node initializes a new session towards the target node, the target node is identified by its ID.

Node Coordinates. The coordinates of node *v* specifies the path from the tree root to *v*. For example if the coordinates of *v* is <0.1.1.2>, then *v* is son number 2 of a node with the coordinates <0.1.1>.

Tree is a structured group of nodes that share the same tree name, which is the ID of the root node. Each node in the tree knows the ID and coordinates of its father and sons. The nodes of the tree and their father-son connections form a legal tree graph.

Tree size presents the number of nodes of the tree. Similarly, every node in a tree holds its sub-tree size.

A Session is a period of time when the two nodes (the source node and target node) exchange speech/voice packets. The Source node initiates the session and either of the parties can trigger the termination of the session. The Source node will try to resume the session in case of a break off. The pair {Source node-ID, Target node-ID} identifies the session uniquely.

Session Path is a bi-directional path used to transfer speech packets between the source node and the Target node. If needed, one or more intermediate nodes will be used to bridge between the session parties. The resources in every node are limited and voice receivers are allocated for a specific session. The voice receivers will be released when the session terminates or when needed, by a forced release. Note that a session path requires radio connectivity between adjacent session path nodes.

Shortcuts are edges used to transfer sessions between two nodes that are within transmission range. The usage method of using shortcuts is described in the metric explained in the introduction. The path allocation algorithm will look for possible shortcuts in order to create a shorter session path.

Figure 6 presents the "natural" session path <0.1.1><0.1><0><0.3><0.3.1><0.3.3.1> and the path created using shortcuts <0.1.1><0.1><0.2.2><0.3.3.1>. The node internal data structures are divided into 2 major parts. The connectivity part keeps updated data describing the node relations with other nodes in the field.

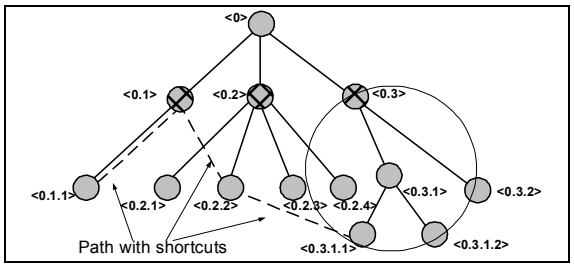

Figure 6: Session path and path shortcuts

The session's part stores the sessions control information. Every session uses a local data structure target node session-cell that keeps the exact state of the session.

2.3. Tree Connectivity Data

Every node *v* broadcasts periodically a hello message. This message carries information about *v*, the coordinates of its neighbors that share the same tree with *v*, and the IDs of the father and sons of *v*. Every node, receiving a hello message from the node *v*, regardless of the source tree, stores/updates the information about *v*.

The IDs are used to validate that v agrees with the receiving node about the type of "relationship" between them. In case that the nodes disagree about the type of relationship for a test-time then the receiving node assumes that something went wrong, and resets the type of the relationship to a "stranger" node.

The following data items are stored in each node to manage the connectivity with the nodes' father, sons and neighbors.

- <Father ID and Coordinates>.
- <Son ID, Coordinates and the size of the son's sub-tree>.
- <Neighbor ID, Coordinates, size of tree and tree name>.

A neighboring node is a node that is within the range and is not a father or son. A neighboring node can be from *v*'s tree or from an adjacent tree. For every neighbor of node *v* that belongs to *v*'s tree, *v* keeps also a list of their neighbors. This information is used for efficient session route allocation (creating of shortcuts).

For tracking the stability of the connection to the father or son, the node counts the number of consecutive times that a Hello message was received from a known father/son but with non-father or non-son indication. Crossing a threshold indicates that there is an inconsistent situation were the father and son or vice versa don't agree on the relations between themselves. In this case, the node resets the connection state to stranger.

2.4. Session Control Data

Every session has the following parties: The Source node who originates the session, the Target node and if necessary intermediate "transit" nodes. Sessions are identified uniquely in the system by the session-id, which is constructed of the Source node and Target node ID's. A session will not break if nodes in the session path change their coordinates. The following data items are used in order to control and maintain the sessions:

A Source node maintains the Target node ID, coordinates of Target node, ID of the next node on the session path and the session state. A Target node maintains the Source node ID and the ID of the previous node on the session path.

Transit nodes maintain the Source node and Target node ID's and the coordinates of previous and next nodes on the session path.

2.5 States

Every node handles two state machines: The connectivity state machine that controls the relations of the node with its neighbors (with the following states: Neighbor addition, Father Loss, Son loss, Son addition, Father Migration, Change of tree size and Son migration) and the session state machine that controls every session (with the following states: Coordinates Resolution, Path Allocation, Stable and Terminating)

2.6 Actions performed by node

Every node takes autonomous actions targeted to keep the node in a tree, and when possible to join a larger tree. When triggered, the node will take actions to create a session to another node. The best connectivity can be achieved only if a session can be created between every two nodes in the field. This requires that all nodes will be organized in the same tree. The following actions are taken as follows to support the maximal connectivity of all nodes in the field: Node Migration (from a small tree to a bigger tree), Node Relocation (inside a tree), Rooting as a tree and registration of a node as a tree.

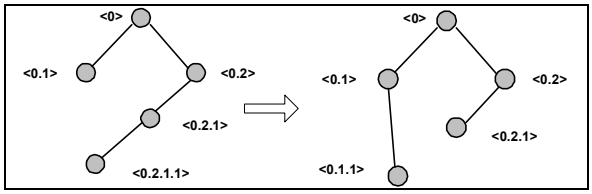

Figure 7: Nodes relocation

Every session occupies a Source node, Target node and if necessary transit nodes. The Source node is the proactive side initiating the session and the Target node and the transit nodes react to the session initiation request. The Source node is in charge of resuming the session after a session break.

3. Simulations and results

We tested the MRA protocol and compared the results to Ad hoc On-demand Distance Vector Routing [3]. AODV minimizes the number of broadcasts by creating routes on demand. For reminding the route discovery process in AODV is initiated whenever a nodes needs to create a session with another node. When possible, nodes use local routing information stored in its tables.

3.1 Simulation environment

We developed a simulator for testing the MRA algorithm and running comparable tests to other routing protocols. Currently the AODV protocol was implemented and used in the tests. A special care was taken during the design and implementation of the simulator on the following aspects: (a) enhanced visualization tools that give a full online view of the testing field, nodes movements, voice channels, specific node status including queues status etc. (b) Tracing the forming of trees in MRA protocol. (c) Tracing the sessions in real time (d) Configuration via online screens (e) Support of logging, debugging and analysis tools.

Our simulator does not fully model the MAC layer. In our simplified model, no packets get lost and transmission reception is granted within the transmission range. Our simplified MAC model allows us to concentrate on the unique features of the MRA algorithm and analyze the results in an isolated environment. This decision allowed us to reduce the number of independent factors and enhance the debug and visualization capabilities.

A session is a full duplex connection between nodes. A message can be lost because of an overflow of the queue in one of the chain of nodes used by the session. The following values were used in the simulations.

Table 1: Main simulation parameters

Attributes	Selected Values
Field Size	120 x 120 meters
No. Of nodes in simulation	100, 140, 180, 220,260 and 300 nodes in the field.
Node Movement	Speed: 5-7 Km/Hour. Movement direction: Change with the probability of 0.01.
Transmission Radius	27 meters.
Session setup	3 retries. The session will be

retries	dropped after the 3rd false retry.
Session Length	5 seconds

3.2 Main Experiments

Table 2 present the number of sessions through which more than a certain percentage of the speech messages passed. For example, the notation MRA 100 presents a run of MRA protocol with 100 nodes. The success rate presents the number of sessions that succeeded to transfer more than X% of the session packets. Naturally, the sessions that succeeded to transfer 85%-100% of the packets are included in 80%-100% column (sessions with less than 80% success of packets transfer were classified as faulty sessions).

Table 2: Successful sessions MRA Vs. AODV

Protocol	Success rate in %				
	80-100	85-100	90-100	95-100	100
MRA100	270	267	264	257	71
AODV100	188	188	188	184	45
MRA140	341	325	315	289	67
AODV140	210	210	210	197	25
MRA180	477	462	444	410	95
AODV180	252	226	172	90	6
MRA220	524	489	452	383	95
AODV220	176	134	81	31	0
MRA260	535	491	441	377	94
AODV260	40	23	12	4	0
MRA300	604	558	518	442	112
AODV 300	13	7	3	0	0

The analysis of the results raises the following observations:

1. MRA generates a higher number of successful sessions. This observation is true for all densities.
2. The gap between the number of sessions generated by MRA and AODV grows as the density of the nodes grows.
3. The session's quality drops quicker for AODV than for MRA as the density increases. The quality of service is defined as the weighted sum of the sessions from every success rate, divided by the estimated attempts. This decrease is mainly noticeable in the high densities. For example, in the case of 300 nodes, MRA succeeded to handle 604 sessions with a success of 80% and AODV handled successfully only 13 sessions. MRA handled successfully 112 sessions with 100% messages transfer and AODV failed to handle any sessions with 100% message transfer.

3.3. Queue overflow

Every node handles 2 types of messages: Control messages which are used in order to control and generate sessions and data messages which are used in order to transfer the speech between the nodes. The total number of messages that a node handles depends on two factors: (1) the number of concurrent sessions managed by the node (Transit sessions or terminating sessions) Naturally, peripheral nodes will transfer less transit traffic than inner nodes. (2) The load generated by the control traffic. This load depends on the density of the nodes and on the nature of the algorithm.

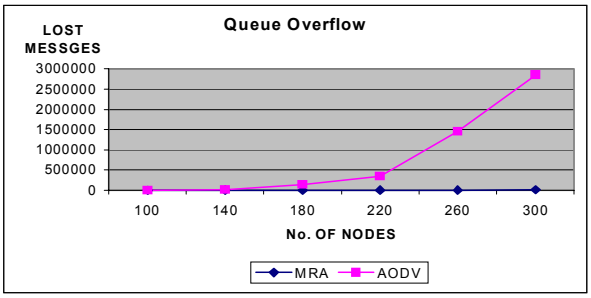

Figure 8: MRA Vs. AODV queue overflow

MRA generates new sessions with an order of O(n) messages while AODV generates sessions with the order of $O(n^2)$. Intuitively, as the number of nodes grows and the density grows, the number of messages handled by a node working under AODV protocol is significantly higher than a node working under MRA protocol. Figure 8 presents the number of lost messages due to queue overflow. The nodes queues are limited to 20 messages. This limit exposes the node to queue overflow and loss of messages. This risk grows as the traffic grows. The number of messages that are lost under AODV grows rapidly compared to MRA. This loss of messages is a central factor decreasing the quality of the session.

3.4. Session Path Length

Table 3 presents the average number of hopes used by the sessions in MRA and AODV. The path created by MRA is shorter except for the exceptional case of 300 nodes where the path created by AODV seems shorter, but in fact AODV has not succeeded to create and maintain a meaningful number of sessions. A shorter path means that less transit nodes are involved in the sessions and the message load occupies fewer nodes. The short path keeps the nodes that are not involved with the session with less traffic and reduces the chance of queue overflow.

Table 3: Session path length

	Number of Nodes in Field					
	100	**140**	**180**	**220**	**260**	**300**
MRA	4.33	4.35	4.27	4.22	4.17	4.1
AODV	4.50	4.39	4.80	4.63	4.4	3.76

4. Conclusions and Future Work

A new type of ad hoc routing algorithm called MRA is presented. The MRA algorithm is designed to maximize a dynamic cover of the mobile nodes by a

network of virtual coordinates. The proposed algorithm covers the nodes by rooted trees and thus establishes the coordinates. The MRA algorithm handles:

- Maximal coverage of the nodes by rooted trees through migration of nodes from smaller trees to larger ones.
- The coordinates are used to forward packets to a destination in spite of sudden disappearances of nodes along selected paths.
- Coordinates are used to find "shortcuts" on the rooted trees so that sub-roots of the trees will not form bottlenecks.
- A distributed "phone" directory for searching the coordinates of nodes is maintained by the algorithm.

The proposed method is compared to the well known AODV (ad Hoc on Demand Distance Vector Routing) algorithm obtaining significant improvements in terms of number of messages, number of concurrent sessions, duration of the sessions, nodes' densities, queue sizes and resilience to sudden movements of the nodes.
Future research directions include the exploration of various regular structures for coordinate coverage as an alternative to the rooted trees presented in this paper. Another direction is the enhancement of the simulator by simulating obstacles and improvements of the simulator performance.

5. References

[1] S. Basagni, I. Chlamatc, V. R. Syrotiuk, B. A. Woodward, "A Distance Routing Effect Algorithm for Mobility (DREAM). ACM/IEEE MobCom 1998

[2] E. Kranakis, H. Singhy, J. Urrutia, "Compass Routing on Geometric Networks", In Proc. 11th Canadian Conference on Computational Geometry, pages 51-54, 1999.

[3] C. E. Perkins, and E. M. Royer. "Ad-hoc On-Demand Distance Vector Routing," Second IEEE Workshop on Mobile Computing Systems and Applications, pp.90-100, February 1999

[4] F. Kuhn, R. Wattenhoffer, A. Zollinger, "Asymptotically Optimal Geometric Mobile Ad-Hoc Routing", In Proc. 6th Int. Workshop on Discreet Algorithms and Methods foe Mobile Computing and Communications (Dial-M), pages 24-33. ACM Press 2002.

[5] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, R. Morris. "A Scalable Location Service for Geographic Ad Hoc Routing," Proceedings of MobiCon, Boston, MA, 2000.

[6] Misra .P, "Routing Protocols for Ad Hoc Mobile Wireless Networks," http://www.cse.ohio-state.edu/~jain/cis788-99/ftp/adhoc_routing/

[7] B. Karp, H. T. Kung. "GPSR: Greedy Perimeter Stateless routing for wireless networks", in Proc. of MobiCom 1998, 2000.

[8] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks," In workshop on Discreate Algorithms and Methods for Mobile Computing and Communications, 1999.

[9] F. Araújo, L. Rodrigues, J. Kaiser, C. Liu, and C. Mitidieri, "A Distributed Hash Table for Wireless Ad Hoc Networks", Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS'05), Columbus, Ohio, USA, June 2005.

[10] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, H. Y. Song, "PARSEC": A Parallel Simulation Environment for Complex Systems Computer," Volume 31, Issue 10, October 1998,Pps: 77 - 85

[11] The ns Manuel, http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf, K. Fall, K. Varadhan, eds. Dec. 2003

[12] A. Rao, S. Ratnasamy, C. Papadimitrou, S. Shenker, I. Stoica, "Geographic Routing withhout Location Information", ACM MobiCom 2003.

[13] F. Kuhn, R. Wattenhofer, Y. Z, A. Zollinger,"Geometric Ad-Hoc Routing: Of Theory and Practice, "in Pricnciples of Didtributed Computing, 2003

[14] D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, New York, NY, USA 1997.

[15] C. Chekuri, S. Kahanna, J. Naor, L. Ziskin, "Approximation Labeling Problem via a New Linear Programming Formulation", Proceedings of the thirtieth annual ACM symposium on Theory of computing, Dallas Texas, 1998.

[16] M.L .Elkin, "A Faster Distributed Protocol for Constructing a Minimum Spanning Tree", in Proc. ACM-SIAM on Discrete Algorithms, SODA'04, pp. 352-361, New Orleans, LA, USA, Jan. 04

[17] J. Eriksson, M. Faloutsos, S. Krishnamurthy, "Scalable Ad Hoc Routing: The Case for Dynamic Addressing"", IEEE INFOCOM 2004