Multi-Path Intelligent Virtual Mobile Nodes for Ad Hoc Networks

Binbin Qian and Jie Wu

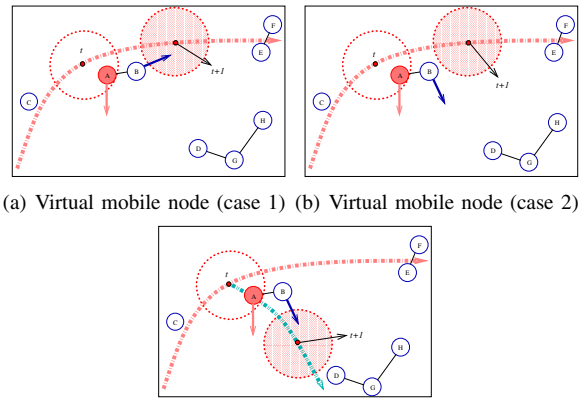
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

***Abstract*—In mobile ad hoc networks, solving the standard problems encountered in fixed networks can be challenging because of the unpredictable motion of mobile nodes. Due to the lack of a fixed infrastructure to serve as the backbone of the network, it is difficult to manage nodes' locations and ensure stable node performance. In this paper, we introduce an extension of an algorithm, Multi-path Intelligent Virtual Mobile Node (MIVMN) Abstraction, which effectively processes the unpredictable motion and availability of mobile nodes in MANETs. Earlier work has applied an abstract node, or virtual mobile node, that consists of a set of real nodes traveling on a predetermined path, or virtual path, which causes unavoidable failure when all the nodes that are emulating the virtual mobile node leave its region. The objective of this paper is to increase the message delivery ratio and decrease the fail ratio of the virtual mobile nodes. In order to achieve this, we allow the messages to switch between multiple Hamiltonian circles. Through simulation results we show that the MIVMN abstraction successfully meets our goals.**

I. Introduction

A mobile ad hoc network (MANET) is a kind of ad hoc network which is composed of a collection of mobile devices equipped with interfaces and networking capability. MANETs are highly dynamic by nature, and the mobile nodes' motions are unpredictable because nodes may fail (leave the network, be powered off, the power is depleted, etc.) and recover again. The set of neighbors for a mobile node can change completely as it moves. Ad hoc networks are a key factor in the evolution of wireless communications. Self-organized ad hoc networks of PDAs or laptops are used in disaster relief, conference, and battlefield environments. These networks inherit the traditional problems of wireless and mobile communications, such as bandwidth optimization, power control, and transmission-quality enhancement. Their multihop nature and possible lack of a fixed infrastructure introduce new research problems such as network configuration, device discovery, and topology maintenance, as well as ad hoc addressing and self-routing [10].

In this paper, we introduce an extension of an algorithm, Multi-path Intelligent Virtual Mobile Node (MIVMN) Abstraction, which effectively processes the unpredictable motion and availability of mobile nodes in MANETs. We run the algorithm on both Real Mobile Nodes (RMNs), whose motions are unpredictable, and virtual mobile nodes - Multi-path Intelligent Virtual Mobile Nodes (MIVMNs), whose motions are predetermined. The MIVMN is extended from the virtual mobile node from [3]. A virtual mobile node is an abstract node with a certain radius moving along a predetermined path, which means it moves in a spatially and temporally consistent manner. At each time step, it is a predetermined distance and direction away from the location it was in the previous time step. Each virtual mobile node is simulated by a constantly changing set of real mobile nodes. The real nodes traveling near the location of a virtual mobile node (i.e. in range of its radius R_V) are allowed to emulate this virtual mobile node. Figure 1 shows different cases of a virtual mobile node routing on the paths. The dashed circle is the virtual mobile node in the current time t, and the shaded circle is the virtual mobile node in time $t+1$. In case 1, the virtual mobile node in the current time t has only one member emulating it in both two time stamps. Node A sends its messages to node B to maintain the state of the virtual mobile node when it leaves the virtual mobile node. In case 2, node B is moving away from the virtual mobile node in time $t+1$, which means the virtual mobile node fails. In case 3, nodes A and B are moving in the same directions as in case 2. However, in this case the virtual mobile node is changing to another path (the blue one) and node B is entering it in time $t+1$, so node A sends the



(a) Virtual mobile node (case 1) (b) Virtual mobile node (case 2)

(c) Virtual mobile node (case 3)

Fig. 1. Different cases of a virtual mobile node routing on the paths.

This work was supported in part by NSF grants ANI 0073736, EIA 0130806, CCR 0329741, CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240. Email: binbinqian@gmail.com, jie@cse.fau.edu

978-1-4244-2390-3/08/$25.00 ©2008 IEEE

virtual mobile node's state to node B.

One of the main motivations of our paper is to reduce the probability of failure. From the above example we see an alternate path can help to avoid an oncoming failure, so in our paper, we allow all the MIVMNs to have multiple paths to choose from. If the MIVMN is going to enter a sparse area, it will choose another path which will probabilistically lead to a denser area. The virtual mobile node can successfully avoid failure by choosing an alternate path that leads to a denser area. Even when all the paths fail, the MIVMN can recover to the initial state when it reenters an area with the minimum allowed density (preset constant).

II. Related Work

There are several previous works that take advantage of geographical information to ease the coordination of mobile nodes. For example, in [7] Nath and Niculesch use a new paradigm called Trajectory Based Forwarding (TBF) to forward packets. TBF is a generalization of source based routing and Cartesian routing. The packets are forwarded by routing packets along a specified curve, which mirrors the topography of the physical surroundings in which it is deployed. The trajectory is embedded in the packet and the intermediate nodes forward packets to the nodes that lie on or next to the trajectory. In our algorithm, the virtual paths are programmed in the mobile nodes, and the current path ID is embedded in the MIVMN's replicas. Instead of finding good trajectories, we supply multiple paths to give the virtual mobile node additional choices when facing potential failure.

The GeoQuorums approach in [4] is also implemented based on the geographic model. Dolev et al. combine the quorum system and geographically local system approaches. In the GeoQuorums algorithm, a focal point abstraction is specified, in which the mobile hosts cooperate to simulate a single virtual process. The network model in their focal point abstraction is static, but in our algorithm, the virtual mobile nodes are all moving on arbitrary, virtual paths. The focal mobile node cannot recover from failure, where as the MIVMN abstraction supports recovery.

In [1] a large amount of redundancy is achieved by distributing a huge number of copies of the data over a geographically local portion of the network. Beal defines a persistent node as a redundant distributed mechanism for storing a key/value pair reliably. The persistent nodes are composed of these atomic particles and the nodes move arbitrarily by shifting their centers. The particles in this paper are synonymous to the real nodes in our model. Our nodes move arbitrarily while the particles remain stationary.

In [6] Lynch et al. describe how a virtual node abstraction can be used to coordinate the motion of real nodes. Similar to our work, their paper describes a communication system used by the virtual nodes. Our paper differs in that our abstraction can be used in sensor networks in which the real nodes move randomly. In [11] Zhao and Tong introduce a connectionless approach to sensor networks. The idea of connectionless networking is to have the capability of moving information between network elements without a preconceived path between source and destination. In our model, since the nodes's movements are unpredictable we do not have a preconceived physical path, but rather a logical one. We also have the virtual mobile node to collect the data and deliver it to the destination when it comes across the destination.

III. Preliminary

The idea of the MIVMN abstraction is largely inspired by [3]. Dolev et al. implement a virtual mobile node by using real nodes to emulate the state of the virtual mobile node using a local communication service.

A. Virtual Mobile Nodes Abstraction

In [3] Dolev et al. define the Virtual Mobile Nodes (VMN) abstraction as applied to MANETs. A service, $Geosensor$, is used to update the mobile nodes' locations and real times. The VMN can be used to solve problems in routing, data collection and other services. For example, for a single VMN case, a real node examines its own location and calculates the virtual mobile node's current location. It then waits until the VMN is nearby and sends the message, or requests to receive messages using the LocalCast service. By increasing the number of VMNs, the messages could also be delivered more rapidly. The VMNs can be directed to systematically explore the region in question, collect and aggregate data, and return necessary data to the clients.

B. Algorithm

In the VMN abstraction, Dolev et al. define *mobile nodes* as either physical mobile nodes or virtual mobile nodes. All mobile nodes support the LocalCast service. A LocalCast service is a local broadcast service which is parameterized by a radius R and delivers messages to all mobile nodes within the specified vicinity; however it may also delay messages being sent or received. The LocalCast service does not distinguish between virtual and real nodes, delivering messages to all mobile nodes that are within its radius R.

Dolev et al. also propose the *mobile point* algorithm to implement the VMN abstraction. A mobile point is defined to be a circular region of radius R that moves along the same path as the virtual mobile node, which indicates that at time t the center of the mobile point is also the location of the VMN. This algorithm consists of a replicated state machine technique: joins, leaves, and recovery. Each real node that resides within the mobile point instantiates the mobile point emulator (MPE) for every VMN. By emulating the VMN, all the real nodes in the VMN carry a replica of the VMN's status. Whenever a real node enters the mobile point's region, it initiates joining the VMN by sending a join request. When it receives the $join\_ack$ message from an active node (the nodes that are emulating the VMN), it starts emulating the VMN by copying the state that is sent together with $join\_ack$ by the active nodes. When a real node leaves the VMN, it stops emulating the VMN and loses its replica of the VMN status.

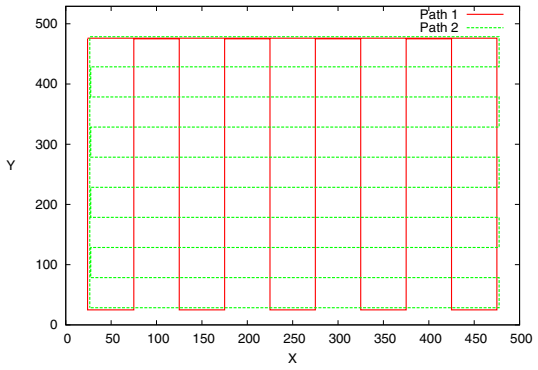


Fig. 2. Two-path layout in a 500×500 square area. The dashed one represents the default path 1. The solid one represents path 2. Path 1 starts from point (25, 25) and winds up to point (25, 475), then goes down back to the starting point. Path 2 starts from point (25, 475) and winds right to the point (475, 475), then goes left back to the starting point.

When the VMN enters an empty region, the VMN encounters a failure.

All the nodes in the network know the location of the VMN, so they can send messages to the VMN as it passes. When the VMN passes by a node i, which is the destination of one of the messages it is carrying, it sends the message that is destined for node i to that node and clears this message on all other nodes that are emulating VMN.

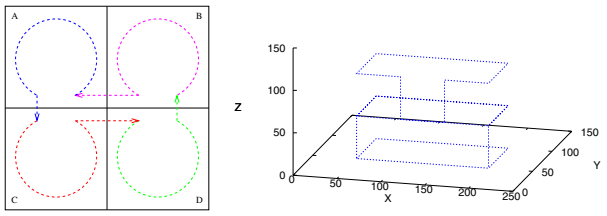
IV. Challenges and Solutions

Since each VMN in [3] only has a single path, it cannot avoid failures. If we add more paths to each virtual mobile node and enable the virtual mobile node to change paths dynamically, then it may avoid oncoming failures. However, adding more paths to the virtual mobile node raises some new problems. First, how to construct the multiple paths? Second, when and how to change paths? Third, how does the recovery mechanism work in the MIVMN abstraction? The solutions of how to solve these challenges are given in the following subsections (B, C and D).

A. Underlying System Model

The MIVMN abstraction lies on top of a physical network in a given region. This network system is composed of both RMNs and MIVMNs. The nodes can be PDAs, for example, carried by people in a section of a city. We use some MIVMNs that travel on two or more virtual paths in this section of the city. The RMNs move in a bounded region of a two-dimensional plane. Each mobile node is assigned a unique identifier from a finite set I. The RMNs travel arbitrarily within the specified region by using the random waypoint model as their mobility model [2] [8].

In the following sections, we use only one MIVMN moving along one of the paths in the network to give a detailed description of a multiple-path implementation. However, in real applications there can be more than one MIVMN to achieve better performance at the expense of higher complexity.



(a) Hierarchical Path (b) 3-D Path View

Fig. 3. (a) A level of hierarchical path. (b) A view of a 3-D path view.

B. Path Construction

Like [3], we use the sevice $Geosensor$ to update all of the nodes' current location and time. $Geosensor$ maintains the current location and real time of each mobile node, (x, y, t). $Geosensor$ can be implemented in a real system by a GPS receiver, or for indoor usage, a Cricket [3][9] device. We assume that the time unit is t_u and the nodes update their locations every time unit.

The motions of RMNs and MIVMNs are completely independent, but a good path design will help the performance of the virtual mobile node. There are a few principles we need to obey when designing paths for our algorithm. First, different paths should complement each other to cover the whole network. Second, all the paths should mainly follow the same direction so that the MIVMN will be able to travel through most of the network even if it changes its path. Last, the physical transmission range should be considered in the design (For instance, in order to keep its status, a virtual node in an intersection should be able to communicate with the RMNs that are outside of its range).

Figure 2 shows an example of a two-path MIVMN implementation. The paths are independent but complement each other by providing the MIVMN a choice of direction at their intersections. The two paths both have a single direction and move in the same general direction, meaning that if one path is moving downward the other path generally moves downward. All the intersections of the two paths except the four corners have two directions, so the MIVMN is able to choose a direction with higher density at the intersections by calculating future densities in the MIVMN region. In this figure, the design of the two paths are the same, but rotated 90 degrees for the other path. The distance (d_i) between two neighboring intersections is decided by the broadcast range.

Figure 3(a) shows a subsection of a hierarchical path. We can divide the region into sections, and the MIVMN travels from the path of one section to the path of a neighboring section. This figure can be a subsection of a larger region which, for example, can be made up of four of these subsections. This kind of path can be applied to large region sizes.

Figure 3(b) shows the view of a 3-D path. There are three planes in this path on which the MIVMN travels. The MIVMN starts from the bottom (XY) plane, and travels along the Z-axis. When the MIVMN traverses the XY plane it goes up to the next plane along the Z-axis, and so forth. When it arrives at the top plane, it changes direction to go back down. While

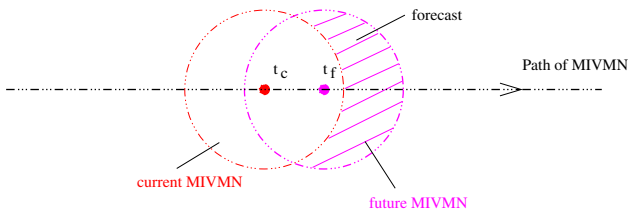


Fig. 4. The left one (solid line) is the MIVMN at current time t_c, and the right one is the MIVMN at t_f ($t_f = t_c + t_u$). The striped region is called $forecast$, which helps to predict the future density of every path.

the MIVMN travels on a plane, it stops at a point which is a distance, R, away from the starting point of the same plane (to ensure that the entire area is covered within the communication radius so the MIVMN is able to go through all areas in the XY plane). By doing so, we get different views of the same path from different angles, for example in this case we get XYZ, XZY, YXZ, YZX, ZXY, ZYX - six path views of the same path coordinates, but with different path patterns. Therefore, a mobile node has six possible path patterns to travel on. We will use the design shown in Figure 2 to implement our algorithm because our goal is not to develop more effective paths, but to demonstrate the performance difference of a single path and multiple path implementation.

C. MIVMN Path Selection

Figure 2 helps describe the solution to challenge two. We use the MIVMN abstraction to enable the virtual mobile node (MIVMN) to choose the best path from multiple Hamiltonian circles when the density falls below a specified threshold by comparing the density of nodes in the MIVMN region on different paths after one time unit t_u. The MIVMN starts from path 1 and checks the future density of both paths at every intersection and chooses the path with the higher density. Since there is only one direction for each path in the corners, the MIVMN is more likely to fail. We can also set one static node in each corner to avoid failure. The next section describes how the MIVMN selects a path from multiple paths.

Figure 4 shows the MIVMN moving in one direction (right in the figure). The striped part in the figure is called $forecast$, which helps to predict the future density of every path after t_u. When an RMN enters the $forecast$, it notifies the MIVMN it is going to join the MIVMN, i.e. it sends a $joinf$ request. Every time the MIVMN receives the notification, it updates the nodes density at a future location on its path (time t_u later). The future location loc_f of the MIVMN can be calculated from current location loc_c as follows.

$$loc_f = loc_c + t_u \times v_{max}$$

where v_{max} is the maximum speed of MIVMN.

In the following, we describe a multiple-path implementation where the MIVMN has a choice of three different paths. We assign unique IDs p_1, p_2, p_3 to paths 1, 2, 3, respectively. We also set p_1 as the default path. In the initial state, the MIVMN moves on the default path p_1. We also define a density threshold $dens_{thr}$ (a constant, i.e. 3 real nodes in the MIVMN region). Whenever it arrives at an intersection of the paths, it checks the current density of the default path. If the future density $dens_1$ at t_u later is lower than $dens_{thr}$, we compare $dens_1$ with $dens_2$ and $dens_3$, where the subscript denotes the path. If either or both of the other path densities is higher than $dens_1$, the MIVMN will switch to the path with the highest density ($dens_2$ if $dens_2$ equals to $dens_3$). Otherwise, the MIVMN remains on p_1.

If the MIVMN is not on the default path p_1 and the future densities of the other two paths are both higher than the current path's and equal to each other, it will choose path p_1. By doing so, the probability to travel on the default path will be higher, which could assure that the MIVMN goes through all real mobile nodes in the region. However, if the densities of all the other paths are lower than or equal to the density of the current one, it stays on the same path.

Since the MIVMN is inclined to change its path when it is approaching a low density area and there is a denser region on another path, the location of the MIVMN cannot be predicted by other real nodes. By routing along the path, the nodes that are emulating the MIVMN carry the path information and broadcast it to other nodes along the paths. However, once the MIVMN changes its path, the old path information becomes invalid. When the MIVMN fails, the MIVMN will not be able to recover successfully as in [3], because none of the other nodes know the direction and location of the MIVMN.

D. Recovery Mechanism

When an RMN enters the MIVMN and cannot communicate with any other active nodes, it broadcasts a reset message. When a node receives a reset message it reinitializes its state. In particular, when the node that discovers the MIVMN has failed by receiving its own reset message, it restarts the MIVMN. However, when the MIVMN resets itself the location and path information stored in the RMNs becomes invalid. In order to maintain the MIVMN and enable it to recover from failures, the MIVMN broadcasts its information (which path it is on and at what time) to the nodes in the future location (one time unit later). Since all nodes update their location every time unit, and the message delivery latency is much smaller than the time unit, the nodes around the different paths should know the current and future location of the MIVMN. A special case also exists: when the MIVMN fails, it will not switch paths at the intersections because there are no nodes in the MIVMN region to predict the future density and to broadcast any change.

V. Implementation

This section describes communication between RMNs and MIVMNs, and how an RMN begins to emulate the MIVMN.

A. Communication Service

In our approach, as in [3], both RMNs and the MIVMN communicate with each other using a local broadcast service. All the real nodes outside the MIVMN (clients) communicate

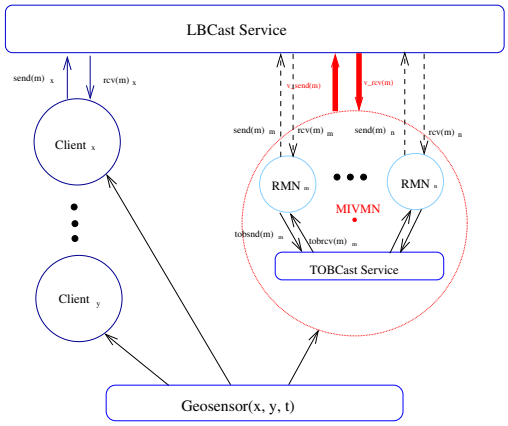


Fig. 5. Local broadcast services and RMN, MIVMN interface.

with each other using a Local BroadCast service (LBCast service) that is characterized by the radius R_R, but the MIVMN and the clients communicate with a local broadcast service that is characterized by radius R_V (which is the range of the MIVMN). The relationship between R_R and R_V is given in the following subsection. Here we use the service module described in [3] to implement the local broadcast services. Figure 5 shows how the local broadcast service interfaces with RMN, MIVMN.

As shown in Figure 5, the clients communicate with their neighbors within a certain range R_R by using the LBCast service. For example, if node i sends a message by performing **send(m)** to any node j that is in range R_R, and is still in range after time delay t_d, then node j will get the message within t_d by performing **rcv(m)**. This model can be implemented in real systems for small R_R[3].

The MIVMN and clients communicate with a local broadcast service that is characterized by radius R_V, we call it V-LBCast. The V-LBCast service differs from LBCast service that is used between clients with radius R_R. As Figure 5 shows the V-LBCast service is implemented with the LBCast service and TOBCast service together.

Since each real node that is emulating the MIVMN carries a replica of the MIVMN, in order to maintain consistency among the replicas in the MIVMN, as in [3], we also use the LBCast service and synchronized clocks to implement a totally-ordered broadcast service (TOBCast service) [3][5]. In this algorithm, a logical clock and physical clock (real time) are used to make sure that events that happened in different nodes are all in order and consistent. Each message contains a time stamp $t_{m,i}$ which equals the time at which the message is sent from i, so if node i sends a message before node j then $t_{m,i} < t_{m,j}$. A time delay of t_d is needed to ensure that the earlier messages are received first. Therefore, the TOBCast service guarantees that messages are delivered in the order in which they are sent (according to real time). If an RMN in the MIVMN region sends a message, then every other real node inside the MIVMN region receives the message in t_d (t_d is much smaller than t_u). Each node can only send a single message for each time unit, when two nodes send the same data in the same logical time unit, we use the nodes' identifiers to break the tie. The node with the numerically smaller identifier wins.

When a client communicates with an RMN in the MIVMN's region, the largest distance between them can $2R_V$. Since a real node only receives updates from the $Geosensor$ every time unit t_u, a real node may be an additional $t_u \times v_{max}$ distance away from the assumed location. Therefore we assume R_R and R_V have the relation as follows:

$$R_R \geq 2R_V + 2t_u \times v_{max} \quad (1)$$

so that the radius of the MIVMN is dependent on the RMNs speed and R_R. With a fixed local broadcast radius, the higher the speed is, the smaller the MIVMN's radius is.

B. MIVMN Emulation

In the MIVMN abstraction, we use a *mobile point emulator (MNE)* to implement the virtual mobile nodes. The mobile point emulator is based on a replicated state machine algorithm that supports joins, leaves, and recovery [5], [3]. Each *MNE* has six states: $inactive$, $pre\_joining$, $in\_forecast$, $joining$, $listening$, and $active$. The state $inactive$ indicates that the RMN is not in the range of MIVMN. The state $pre\_joining$ indicates that the real mobile node is in the $forecast$ and waiting for the ACK message from the MIVMN and state $in\_forecast$ indicates a received ACK message from the MIVMN. The states $joining$ and $listening$ mean the RMN is in range of the MIVMN and in process of joining the MIVMN. The state $active$ means that the RMN is emulating the MIVMN.

Whenever real mobile node i enters the $forecast$ region, it sends a message $joinf$ to MIVMN, which includes the ID of node i and path number p. Then node i changes to the $pre\_joining$ state and waits for the $joinf\_ack$ from the MIVMN. Any active node j that receives the request broadcasts the notification to every other member in the MIVMN through TOBCast service and updates the nodes density of the MIVMN region ($dens$) at time t_f (t_u later than the current time). Since each node ID is unique, it is ensured that the nodes in the MIVMN only calculate the same node once. Since the paths are programmed in the real mobile nodes, the virtual mobile node is able to predict the future density of different paths. When some active node receives a message $joinf$, it sends an acknowledgment, $joinf\_ack$. When node i receives the acknowledgment, its state changes to $in\_forecast$. When a real mobile node i enters the MIVMN region, it sends the $join\_req$ to the MIVMN, which includes the identifier of node i. Then, it start its $joining$ state. When it receives its own join request through TOBCast, it enters the $listening$ state. In this state, node i is able to monitor the messages in the MIVMN and save the messages that it cannot yet process. Any active node j that receives the request sends a $join\_ack$, which includes a copy of the state of MIVMN and the identifier of node i. Then node i enters the $active$ state and starts emulating the virtual mobile node.

TABLE I
SIMULATION PARAMETERS

Parameters	Value	Parameters	Value
RMN Pop Size	100-200	Region Size	500×500
RMN Speed	15-20	Sim Time	4000
PauseTime	2.0-5.0	Time unit (t_u)	0.5
LBCast Radius R_R	100	Delay (t_d)	0.1
MIVMN Speed	15-20	Path Qty	2
Msg Buffer Size	100	Density threshold	3

VI. SIMULATION

The goal of our paper is to demonstrate that by implementing the MIVMN on multiple paths, the failure ratio will be decreased and the message delivery ratio will be higher than the single-path virtual mobile node implementation [3]. In this section, first we present the simulation settings. Then we analyze the implementation results to prove the reliability of the MIVMN.

A. Simulation Settings

We use a custom simulator to implement our algorithm. It is written in the C++ programming language using the object-oriented model. We simulate a wireless ad hoc network in a square region, like a section of a city. We distribute a set of real mobile nodes to compose a dense network. All the real mobile nodes in the network move randomly by using the random waypoint mobility model and the virtual mobile node moves on one of two predetermined virtual paths (the two paths are constructed as in Figure 2). We use different objects that interface with each other. The network has an *LbCast* service object, a *TOBCast* service object, and a set of *node* and *mne* objects. Every node in the physical network is represented by both a *node* object and *mne* object. We implement our algorithm as event-driven: updating current location and time, sending messages, and receiving messages.

Since our focus is to demonstrate the improvement of adding one more path, and the time delay t_d (0.1 logical time) is supposed to be much smaller than the time unit t_u (0.5 logical time), we assume that when a node enters the MIVMN, it changes its state to $active$ and begins to emulate the MIVMN in the same time unit. This is because a node (including both RMNs and the MIVMN) only updates its location every time unit, and in one time unit, the node has enough time to send a join request and be acknowledged. We also assume all the nodes know the location of the MIVMN in our simulation.

Our simulation is based on logical time, and each time unit t_u is 0.5 logical time. The simulation settings are shown in Table I. We always count the time delay as the worst case, which means we count every time delay as the maximum time delay (0.1 logical time in our simulation). When the message buffer is full, the node will drop the oldest message and push the new one to the back. Each node (including both the MIVMN and the RMNs) updates its location every time unit. To ensure that all of the messages that are generated are delivered, every 5 logical time, we let the node choose whether or not to generate a message with probability 0.15. After $1/3$ of the whole simulation time, the nodes stop generating any messages. By doing so, we can ensure that there is enough time to deliver all of the messages generated.

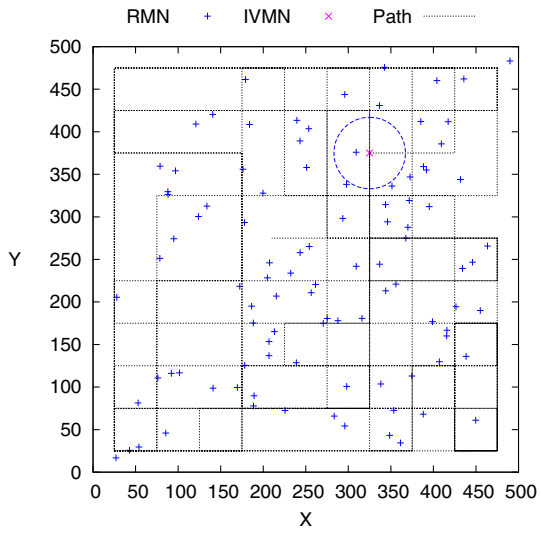


Fig. 6. Nodes' distribution (both RMNs and MIVMN) at one point in the simulation with population size 100 for the first 1000 simulation time run. The dashed circle around MIVMN is the MIVMN range.

TABLE II
MIVMN REGION DENSITY ON CURRENT PATH AT DIFFERENT TIME

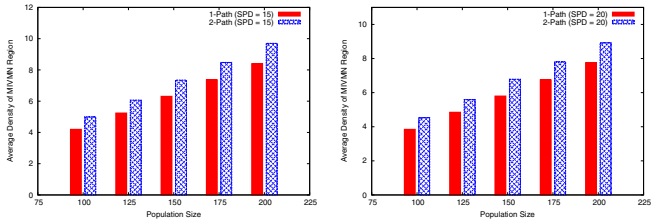
Time	110	110.5	**111**	111.5	112	112.5	113	113.5
Path #	1	1	**1**	2	2	2	2	2
Density	3	2	**1**	3	3	3	4	5

B. Simulation Results and Discussions

For each tunable parameter, our simulation is repeated until the confidence interval is sufficiently small (less than $\pm 5\%$, for the confidence level of 90%).

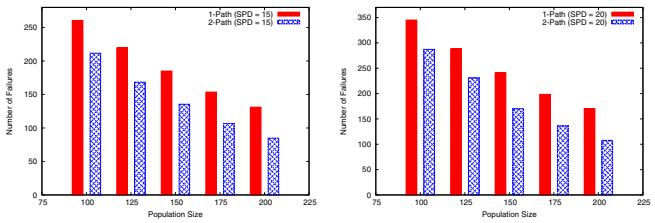
Figure 6 shows the distribution of all nodes (both MIVMN and RMN) in the region at logical time 111 of one run, where the MIVMN is about to switch from path 1 to path 2 in the next time unit. It also shows the course the MIVMN took, switching between path 1 and path 2 in the first 1000 logical time units. Since the current density (2 nodes per MIVMN range for the current time unit) is smaller than the threshold (3 nodes per MIVMN range), the MIVMN switches to the alternate path. If the MIVMN stays on path 1, it is going left. If the MIVMN switches to path 2, it is going downward. From the distribution in this time unit, we can also tell that the future density of the MIVMN region on path 1 is smaller than the density of the MIVMN region on path 2. So the MIVMN switches its current path from path 1 to path 2. By doing so, the failure ratio is reduced. Table II shows the variance of the density of the MIVMN region on the current path from time 111 to 113.5 in the same run as Figure 6. From the table we can see the density was decreasing on path 1 until it switched to path 2.

Figure 7 shows, for speed 15 and speed 20, the average density of the MIVMN region with different population sizes



(a) Average density of SPD 15 (b) Average density of SPD 20

Fig. 7. Average density of the MIVMN region with different population size in speed (SPD) 15 and speed (SPD) 20.



(a) The number of failures (b) The number of failures

Fig. 8. The number of failures of the MIVMN with different population sizes at (a) speed 15, (b) speed 20.

TABLE III
AVERAGE DENSITY OF MIVMN REGION AT SPEED 15 AND 20 WITH CONFIDENCE LEVEL 90% (P# - PATH NUMBER)

POP	P#	Density AVG		C1		C2	
		Spd15	Spd20	Spd15	Spd20	Spd15	Spd20
100	1	4.20	3.85	4.16	3.81	4.25	3.89
125	1	5.24	4.85	5.21	4.83	5.28	4.87
150	1	6.31	5.80	6.26	5.77	6.37	5.83
175	1	7.37	6.75	7.31	6.72	7.42	6.79
200	1	8.40	7.76	8.34	7.72	8.45	7.79
100	2	4.99	4.53	4.93	4.48	5.05	4.57
125	2	6.07	5.59	5.98	5.53	6.16	5.65
150	2	7.35	6.79	7.28	6.73	7.43	6.85
175	2	8.48	7.81	8.37	7.74	8.58	7.88
200	2	9.70	8.92	9.60	8.85	9.80	8.99

TABLE IV
THE NUMBER OF FAILURES OF THE MVIMN AT SPEED 15 AND 20 WITH CONFIDENCE LEVEL 90% (P# - PATH NUMBER)

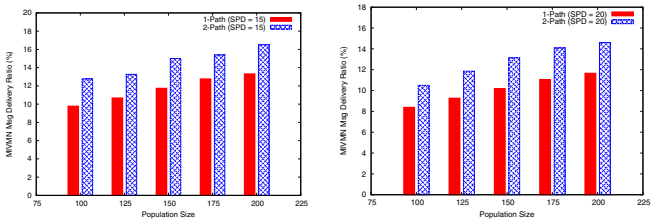
POP	P#	No. Failures		C1		C2	
		Spd15	Spd20	Spd15	Spd20	Spd15	Spd20
100	1	261	345	256	341	265	349
125	1	219	288	215	284	224	292
150	1	185	241	180	235	189	246
175	1	154	198	149	194	158	203
200	1	131	170	127	165	135	175
100	2	212	287	207	282	216	293
125	2	168	231	164	225	173	236
150	2	136	170	132	166	139	175
175	2	107	136	102	132	111	140
200	2	85	108	81	104	89	112

(100, 125, 150, 175 and 200 nodes). For both speeds 15 and 20, the average density of the two-path MIVMN implementation is always higher than the one-path implementation. Even though the difference in density is not big, one node could make a big difference. The density increases when the population size grows because the nodes have a much higher chance to emulate the MIVMN. Comparing Figure 7(a) with Figure 7(b), we can see that the average density of the MIVMN region at speed 15 is always higher than at speed 20 with the same population size. That is because for a higher speed, the MIVMN's radius will be smaller. As the speed increases, the amount of time for which the real nodes and the MIVMN are in range of each other decreases. Table III shows the values of average density of the MIVMN and that the confidence interval at speeds 15 and 20 is around $\pm 1\%$.

Figure 8 shows that the number of failures of the MIVMN is lower than the single-path implementation. The decrease of the number of failures as the population size increases for both speeds is similar. Since the number of failures with a higher population size is smaller, the MIVMN performance is more significant with higher population size. Comparing 8(a) and 8(b), we can see that the number of failures of the MIVMN at speed 20 is much higher than at speed 15. This is because the MIVMN is more stable at a lower speed. Table IV shows the number of failures of the MIVMN at speeds 15 and 20 and that the confidence interval for the average delivery ratio is less than $\pm 5\%$.

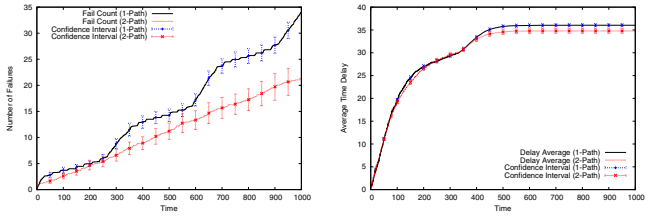
Since almost all of the messages that are generated are successfully delivered, we calculate the percentage of the messages that are delivered by the MIVMN of both single-path implementation and two-path implementation, through which we can show the performance of MIVMN. Figure 9 shows the MIVMN message delivery ratio with different population sizes at both speeds 15 and 20. From this figure we can tell that by adding another path, the MIVMN message delivery ratio increases. Also, for different population sizes, the MIVMN's performance is stable. When the population grows, the density of the MIVMN region grows, which helps deliver more messages. Comparing Figure 9(a) with Figure 9(b), we can see that the MIVMN delivery ratio at speed 15 is generally better than at speed 20. This is due to the same issue that causes the density of the MIVMN region at speed 15 to be higher than at speed 20.

Figure 10 shows that by adding one more path, the performance of the MIVMN, including both delivery ratio and time delay of message delivery, is better than with a single path. Figure 10(a) shows that the number of failures in both scenarios increases with time, but the 2-Path MIVMN implementation fails less than the 1-Path MIVMN implementation. Figure 10(b) shows that the time delays in both scenarios increase with time. The time delay for 2-Path MIVMN implementation is generally lower than 1-Path MIVMN implementation. This



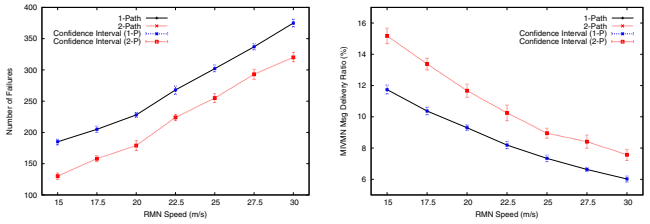
(a) Delivery ratio of SPD 15 (b) Delivery ratio of SPD 20

Fig. 9. The MIVMN delivery ratio with different population size at (a) speed 15, (b) speed 20.



(a) Number of failures (b) Average time delay of message delivery

Fig. 10. The number of failures of the MIVMN along time at (a), the time delay of message delivery along time, with a confidence level of 90% at (b).



(a) Number of failures (b) MIVMN message delivery Ratio

Fig. 11. (a) The number of failures of the MIVMN with different RMN speed, (b) The MIVMN message delivery ratio with different RMN speed.

is because when the time increases, the MIVMN changes path more frequently and more failures occur, which may increase the average delay of message delivery. After a certain time run, the time delay converges. This is because the nodes stopped generating messages and finished delivering the ones they had.

In order to make sure that RMNs inside the MIVMN range can transfer messages to an RMN outside of range, we set the MIVMN's speed less than or equal to the RMN's. Figure 11 shows the MIVMN message delivery ratio and the number of failures of the MIVMN with a population size 150 to give us an idea of the MIVMN's performance when the RMNs raises their speed. The MIVMN speed is $15m/s$ and the RMNs travel at different speeds (15.0, 17.5, 20.0, 22.5, 25.0, 27.5 and 30.0). The MIVMN's density threshold is set to 3. Figure 11(a) shows that the MIVMN fails twice as many times when the RMN move twice as fast. This is because the duration of time in the MIVMN region becomes shorter when the RMNs have a higher speed. For example if the MIVMN and RMN i are moving in the same direction and RMN i moves at speed 15, then node i will stay in the MIVMN's range until either one changes direction. However, if node i moves at speed 30, then it will leave the MIVMN's range at less than or equal to $2R_V/15$ time units. In conclusion, the MIVMN message delivery ratio is mainly impacted by the number of failures. Figure 11(b) shows that the MIVMN message delivery ratio is twice as low when the RMNs move twice as fast. Thus here we recommend a similar speed for both RMNs and the MIVMN.

C. Simulation Summary

Concluding all of the simulation results, we see that by adding one more virtual path in our algorithm, we get a better performance of the MIVMN. Even though the computation complexity is raised by the additional path, the results show that the number of failures is decreased and the MIVMN is able to deliver significantly more messages with similar or even shorter time delays. We believe that adding additional paths to the MIVMN will help to advance the performance even more, but we also need to know that the number of paths needs to be limited to avoid the problem of diminishing returns. That is, as the cost (complexity) increases, the amount of benefit gained decreases until the cost far outweighs the benefit. The benefit refers to a decrease in the failure rate and an increase in the message delivery rate.

VII. Conclusion

In this paper, we have analyzed the challenge of the highly-dynamic nature and the unpredictable motion of MANETs. Our work uses virtual mobile nodes (MIVMNs) to traverse through an entire network by traveling on multiple virtual Hamiltonian circles. The MIVMNs collect all the data from other nodes near the paths and deliver them to the destination when they meet. Our focus is on enabling the MIVMN to have a choice to switch between multiple paths instead of moving on a single path. Our work successfully reduces the number of failures, thus increasing the message delivery ratio. In the future, we plan to include the choice of more paths and more virtual mobile nodes that work together to increase delivery ratio. Of course, the inclusion of more MIVMNs and paths will increase the complexity of the abstraction, like RMNs' power consumption vs. MIVMN's failure rate decrease. Thusly, some future work may involve finding an equilibrium between the overhead produced by the additional complexity and the gain in throughput.

References

[1] J. Beal. Persistent nodes for reliable memory in geographically local networks. Technical report, AIM-2003-011, MIT, April 2003.

[2] T. Camp, J. Boleng, and V. Daives. A survey of mobility models for ad hoc network research. *Wireless Communication and Mobile Computing (WCMC)*, 2(5):483–502, September 2002.

[3] S. Dolev, S. Gilbert, N. Lynch, E. Schiller, A.Shvartsman, and J. Welch. Virtual mobile nodes for ad hoc networks. In *Proc. of the International Conference on Principles of Distributed Computing (DISC)*, 2004.

[4] S. Dolev, S. Gilbert, N. Lynch, A. Shvartsman, and J. Welch. Geoquorums: Implementing atomic memory in ad hoc networks. Technical report, LCS-TR-900, MIT, 2003.

[5] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[6] N. Lynch, S. Mitra, and T. Nolte. Motion coordination using virtual nodes. In *Proceedings of 44th IEEE Conference on Decision and Control (CDC05)*, Seville, Spain, December 2005.

[7] B. Nath and D. Niculescu. Routing on a curve. *ACM SIGCOMM Computer Communication Review*, 33(1):150–160, 2003.

[8] S. PalChaudhuri, J. Boudec, and M. Vojnovic. Perfect simulations for random trip mobility models. In *Proc. of the 38th Annual Simulation Symposium*, April 2005.

[9] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proc. of the 6th MobiCom*, 2000.

[10] J. Wu and I. Stojmenovic. Ad hoc networks. *IEEE Computer*, 37(2):29–31, Feburary 2004.

[11] Q. Zhao and L. Tong. A connectionless approach to large scalesensor networks. In *Proc. of IEEE MILCOM'04*, 2004.