
Autonomous Agents and Multi-Agent Systems, 8, 47–68, 2004

MARP: A Multi-Agent Routing Protocol for Mobile Wireless Ad Hoc Networks

ROMIT ROY CHOUDHURY croy@uiuc.edu
Dept. of Electrical and Computer Engineering, University of Illinois at Urbana Champaign, USA

KRISHNA PAUL krishna@it.iitb.ac.in
School of Information Technology, Indian Institute of Technology, Bombay

SOMPRAKASH BANDYOPADHYAY somprakash@iimcal.ac.in
Dept. of Management Information Systems, Indian Institute of Management, Calcutta

Abstract. Supporting mobility in a multi hop wireless environment like the MANET still remains a point of research, especially in the context of time-constrained applications. The incapacity of ad hoc networks to offer services of the likes of static or infrastructured networks may be attributed to two major reasons. One, unpredictable mobility of hosts cause location-transparent-packet-delivery to be implemented only at the expense of large control overhead. Two, the lack of central control causes connection management and scalability to be major problems in the multi hop environment. In this paper we propose an efficient agent based routing mechanism that not only incurs minimal overhead, but also lays the foundation for additional functionalities as network management and real time applications. In other words, we show that the agent framework makes the MANET robust and survivable under stringent system constraints.

Keywords: mobile agents, ad hoc networks, topology awareness.

1. Introduction

Ad hoc networks [3–5] are envisioned as infrastructure-less networks where each node is a mobile router, equipped with a wireless transceiver. A message transfer in an ad hoc network environment could either take place between two nodes that are within the transmission range of each other or between nodes that are indirectly connected via multiple hops through some intermediate nodes. This implies that the nodes, which act as intermediate nodes in the data transfer process, must be willing to participate in communication until successful message transfer has been accomplished. The failure of such an event would amount to messages getting lost or message transfer getting interrupted. The dynamics of wireless ad hoc networks, as a consequence of mobility and disconnection of mobile hosts, pose a number of problems in designing proper routing schemes for effective communication [3, 7–9]. To maintain a session between two nodes for a long span of time, the caller node needs to be aware of the frequently changing route status, and subsequently, of newly available routes. This points to a form of topology awareness[18] that either should be incorporated proactively or on-demand.

In the first part of our work, we have devised an agent-based framework with its associated protocols and mechanisms. The agents in the framework move from one node to another, giving and taking relevant information, with the primary objective of making

all nodes in the system, *topology-aware*. This topology awareness is used in the context of establishing and maintaining a communication link between two nodes. Put differently, a node periodically receives a refreshed view of the network, enabling it to constantly evaluate network conditions. This periodically-updated network information may be used as a knowledge-base, based on which intelligent decisions like adaptive route selection, etc. could be made. The behavior of the knowledge-base (past and present) could be analysed and extrapolated to predict network behavior in the near future. For example, a successful prediction may result in forseeing route errors. Doing the needful in such scenarios could improve network performance significantly. More importantly, since the knowledge-base is maintained in the local cache of each node, decision-making could be autonomous and distributed. Thus it seems that a foundation could be laid for the network to offer services of the likes of static networks. In the second part of our work we study the survivability issues of the Ad Hoc network under severe constraints and the advantages of incorporating the agent system against conventional non-agent systems. In our results we see that, although proactive agent navigation incurs network overhead, the amortized cost of supporting agents proves to be beneficial when considering its numerous advantages.

2. Related work

In this paper we propose to introduce an agent-based system into the network architecture. Although there has been substantial work on *mobile software agents*, its applicability in wireless ad hoc environments has received limited research attention. Protocols in ad hoc networks have mostly comprised of non-agent systems. In this section we discuss some of the earlier works in both agent and non agent systems in ad hoc environments.

The conventional proactive routing protocols that require to know the topology of the entire network is not suitable in such a highly dynamic environment [9], since topology updates need to be broadcast frequently throughout the network. These update-packets consume a large portion of the network bandwidth, even when network traffic is low. Using the *Multi-Agent Routing Protocol* (MARP), we show how the overhead associated with routing is minimal, and the possibility of adaptively controlling this overhead according to traffic conditions.

In contrast to a proactive mechanism, a demand-based, reactive route discovery procedure generates a large volume of bursty control traffic. The actual data transmission is also delayed until the route is determined [6]. In addition, route rediscovery (in the event of route errors) consume a considerable amount of time; resulting in violation of ‘‘delay constraints’’ in packet delivery. MARP in comparison, exhibits a much stable behavior, with negligible latency in reacting to route failures. In summary, the proactive and reactive routing mechanisms perform well only under certain traffic conditions. For example, the DSR routing protocol might perform route rediscovery too often during a real time communication – consuming useful data bandwidth to transmit *route requests, route replies* and *route errors*. MARP, as we show later, do not suffer similar problems and may thus be suitable over a larger operating region.

In [6], a preemptive route discovery has been proposed in order to discover best routes dynamically and then adaptively use them for continuing communication. However,

the route-discovery mechanism recurringly floods control packets when a route is stable. This causes unproductive traffic in the environment and may thus increase end-to-end delay for communication. Also, loss of control packets would mean the unavailability of stable routes and may interrupt the ongoing process of communication. In [10], a reactive route discovery has been performed in which the caller node broadcasts a control packet at the event of a communication request. In a large system the congestion increases exponentially during this procedure and may prove to be detrimental in a high-traffic network. Re-broadcasting control packets in the event of losses, aggravates the situation further. In our proposed routing protocol, the network enjoys the ability to perform a form of preemptive routing while maintaining low control overhead.

There are proposals to reduce control traffic generated in reactive protocols. For example, the Location-Aided Routing (LAR) Protocol [11] suggest approaches to decrease overhead of route discovery by utilizing location information for mobile hosts. But these proposals assume the support of Global Positioning Systems (GPS) for information on the geographical location of mobile hosts. Additionally, the LAR protocol assumes that the correct location information about the intended destination node is available to the caller node before it determines the expected zone. In contrast, our agent-based system functions without the support of GPS.

Agent based approaches for information management and routing have been evaluated in [18, 19, 21]. Studies in these papers show that using the agent paradigm in a wireless network may be significantly beneficial. However, most of the approaches focus on the distribution and collection of information in static networks. In this paper we investigate agent application in highly mobile ad hoc networks. In addition, the agent based system we propose, provides room for generic and adaptive decision-making based on network conditions. As an example, an agency may reduce its agility when the network traffic is observed to be low, or when battery power needs to be conserved.

3. Problem formulation

Conventional approaches towards issues related to mobile multi-hop environments have suffered major drawbacks in the context of supporting data communication:

1. Extensive exchange of control packets (mostly in proactive mechanisms) to continuously track mobile hosts are often unnecessary and add considerably to the load on the network.
2. Frequent route errors, and equally frequent route re-discoveries (in reactive techniques), violate delay constraints in real time data communication. Route re-discoveries also involve network flooding, implying congestion.
3. Different types of control packets, each catering to only the instantaneous needs of individual hosts have caused sub-optimal consumption of the network bandwidth. Some new notions of opportunistic routing and piggybacking have attempted to resolve this problem.

In short, an essential crisis in MANETs lie in the difficulty of supporting distributed multihop communication (often requiring to be uninterrupted and delay constrained) over

a dynamic and unpredictable topology, while keeping the control traffic below reasonable bounds.

The motivation for this paper is drawn from the above problem statement. We have addressed the crisis of carrying out a communication, uninterrupted, unto its accomplishment. Of course the agent protocol adaptively ensures low control traffic and almost eliminates the delay involved in switching between routes in the event of route errors. In other words, through our agent protocol, a node is always aware of multiple paths in the spatial domain. In addition, we also suggest extensions to this agent-based protocol to incorporate features of real time support and load balancing.

Thus on a whole we show that the agent based framework performs better in comparison to other protocols when it comes to a question of connectivity, latency, congestion and network adaptation. In section IV we define some of the terms that we use in this paper. Section V discusses the agent framework and the agent navigation strategies. Section VI describes the information exchange and location prediction mechanisms. In Section VII, we discuss the simulation model and explain the potential advantages of such a model. Section VIII discusses issues related to data communication using MARP and Section IX presents a comparative performance analysis of agent and non agent systems. Section X discusses some of the issues and insights from our simulations. We summarize the paper in Section XI, with a brief conclusion.

4. Preliminaries and relevant terms

4.1. Affinity and stability

Affinity a_{nm}, associated with a link l_{nm}, is a prediction about the life span of a link between nodes n and m [10]. For simplicity, we assume bidirectional links, implying a_{nm} to be equal to a_{mn}. Also, let the transmission range be R. We would later point out how this assumption could be relaxed without affecting the correctness of our protocol. To find out the affinity a_{nm}, node n sends a periodic beacon and node m samples the strength of signals received from node n, periodically. Since the received signal strength, S, varies inversely with the square of the distance, d, between the transmitter and the receiver (in open ground), it may be possible to conservatively predict d from S. If M is the average velocity of the nodes, the worst-case affinity a_{nm} at time t is (R-d)/M, assuming that at time t, the node m has started moving away from n with average velocity M. For example, If the transmission range is 300 meters, the average velocity is 10m/sec and current distance between n and m is 100 meters, the life-span of connectivity between n and m (worst-case) is 20 seconds, assuming that the node m is moving away from n in a direction obtained by joining n and m.

It is well known that the lifetime of a path is equal to the lifetime of the weakest link in that path. Thus, given a path p = (s, i, j, k . . . l, d), the *stability* of p [10] at a given instant of time may be defined as the lowest-affinity link contained in that path at that instant of time. Formally, stability of path p, η^p, between two nodes s and d, is as follows:

$$\eta^p = \min\ [a_{si}, a_{ij}, a_{jk} \ldots a_{ld}]$$

However, the notion of stability of a path is dynamic and context-sensitive. As indicated earlier, stability of a path is the life-span of that path, from a given instant of time.

However, stability must be viewed in the context of providing a service. A path between nodes *s* and *d* may be considered stable if the lifespan of that path is sufficiently long to accomplish transfer of a specified volume of data, from *s* to *d*. Hence at a particular instance of time, a path that is stable for a given flow, may be unstable for a different flow although both flows are between the same sender and destination nodes.

4.2. Recency

One of the aspects that make mobile ad hoc environments significantly different from static or centralized environments is that topology information gets stale with time, in the former. This means that any information that a node A receives regarding some other node B in the mobile network is only partially correct (since there is at least a difference of propagation delay between the procurement of the information from node B and its delivery to node A). This implies that any information must thus be recognized with a degree of correctness, i.e., if node A now has two different information regarding node B, it must have the capacity to accept only the one which is more correct; to be more precise, information that is more recent.

In the context of our protocol, let us assume that two agents A1 and A2 arrive at node *n*, both of them carrying information about node *m* which is multi-hop away from node *n*. In order to update the topology information at node *n* about node *m*, there has to be a mechanism to find out who carries the most recent information about node *m: agent A1 or agent A2*?

To solve this problem, every node in the network maintains a counter that is initialized to 0 when the network commences. We term this counter as *recency token*. As we see later, agents jump form one node to another collecting and distributing network information to nodes. Now, when an agent has completed its tasks and is about to jump away from a node, it increments this recency token counter by one and stores the new value against that node's ID within its own data structures. Obviously, at any given instance, the magnitude of the recency token of any node represents the number of times that node was visited by agents since the commencement of the network. This also implies that if two agents have a set of data concerning the same node, say node m, then the agent carrying the higher recency token value of node n has more current information about it. We discuss the implications of recency tokens in detail, in later sections of this paper.

4.3. Time to migrate (TtM)

An agent visiting a node is not allowed to migrate immediately to another node. An agent will be forced to stay in a node for a pre-specified period of time, termed as *time-to-migrate* (TtM), before migrating to another node. By controlling TtM, the network congestion due to agent traffic can be controlled. For example, if TtM = 100 msec, for a single-agent system, it implies that the wireless medium will see one agent in every 100 msec. In our simulation, it has been assumed that an agent would take approximately 3 msec. to physically migrate from one node to another. So, in this example, the wireless medium would be free from agent traffic 97 percent of the time.

On the other hand, reducing the agent traffic (by increasing TtM) reduces the frequency at which agents may visit network hosts. This may prove to be unsuitable in a highly

mobile system where topology changes at a fast pace. The trade off is thus between congestion and convergence.

4.4. *Average connectivity convergence*

We have developed a metric, *average connectivity convergence*, to quantify the deviation between *actual network topology* and the *network topology perceived by individual nodes* at any instant of time.

Let l^a_{nm} be the binary link status (0 for disconnectivity and 1 for connectivity) between nodes n and m as perceived by node a at any instant of time. Let l_{nm} be the actual link status between node n and m at the same instant. Information about link status l^a_{nm} is said to converge at node a, iff $l^a_{nm} = l_{nm}$. Thus, connectivity convergence of a link between n and m at node a, $\gamma^a_{nm} = 1$, if $l^a_{nm} = l_{nm}$ and 0 otherwise. Connectivity convergence of node a, γ^a, for all links in a network of N nodes, is defined as:

$$\gamma^a = \frac{\sum_{forallnode-pairs-ij}(\gamma^a_{ij})}{N \times (N-1)/2}.$$

where, $N \times (N-1)/2$ denotes the total number of node pairs in the network. At a given time, if γ^a equals 1.0, it implies that the connectivity information at node a is exactly the same as the actual network connectivity at that given time. As another example, in a 10-node network, there are 45 node-pairs and 45 possible link-status. If, at any node a, 44 link-status' match (at any instant of time) with the actual link-status', then $\gamma^a = 44/45 = 0.98$.

The *average connectivity convergence* for the network is defined as

$$\gamma_{avg} = \frac{\sum_{forallnode-k}(\gamma^k)}{N}.$$

4.5. *Average link-affinity convergence*

Average connectivity convergence quantifies the deviation of actual network topology with the network topology perceived by individual nodes, in a discrete manner (where link status is 0 for disconnectivity and 1 for connectivity). However, if we can quantify link status based on link-affinity, the quantification could be more appropriate in formulating a metric, which would help us to evaluate the difference between the actual network topology and the network topology as perceived by individual nodes in a continuous scale.

Let α^a_{nm} be the affinity between node n and m as perceived by node a at any instant of time and α_{nm} be the actual affinity between node m and n at the same instant. Information about link status α^a_{nm} is said to converge at node a, iff $\alpha^a_{nm} <= \alpha_{nm}$. As indicated earlier, affinity is a worst-case prediction about the lifespan of a link. So, if the affinity of a link between n and m as perceived by a node a is less than actual affinity between n and m, we accept the perception of node a about the link-affinity between n and m. However, if $\alpha^a_{nm} > \alpha_{nm}$, we will deem this as over-estimation of affinity at node a and call the perception incorrect.

Thus, *link-affinity convergence* of link between n and m at node a, $\lambda^a_{nm} = 1$, if $\alpha^a_{nm} <= \alpha_{nm}$ and 0 otherwise. *Link-affinity convergence of node a*, λ^a, for all links in the network, is thus defined as:

$$\lambda_a = \frac{\sum_{forallnode-pairs-ij}(\lambda^a_{ij})}{N \times (N-1)/2}$$

At some instant of time, if $\lambda_a = 1.0$, it implies that the topology information at node a is 100 percent acceptable, so far as affinity-based prediction mechanism is concerned.

The *average link-affinity convergence* λ_{avg} for the network is thus defined as

$$\lambda_{avg} = \frac{\sum_{forallnode-k}(\lambda^k)}{N}$$

5. Mobile agents

Mobile agents [19, 20] are a novel effective paradigm for distributed applications, and are particularly attractive in a dynamic network environment involving partially connected computing elements [1, 2]. The notion of computation mobility against conventional data mobility governs the underlying philosophy of agencies. Most research examples of the mobile agent paradigm as reported in the current literature have two general goals: reduction of network traffic and asynchronous interaction. Some authors have suggested that agents can be used to implement network management [13, 14] and to deliver network services [15]. Intensive research on the ''Insect-like Systems'' has been done over the last few years. The mobile agent systems have been popularly simulated in close resemblance to an ant colony [12, 16, 17]. Of particular interest is a technique for indirect inter-agent communication, called *stigmergy*, in which agents populate information cache of nodes, which other agents can use. The technique of overwriting a set of information with more appropriate information has been popularly called *blackboard communication* [16, 22]. Stigmergy serves as a robust mechanism for information sharing. In our protocol, we have used a multi agent framework that incorporates stigmergy for mutual interaction. The blackboard form of communication has also been implemented for agent-node communication.

5.1. Issues in implementing the agent paradigm

Why not have a single agent? The topology traversing could well be performed using a single agent with a suitably low TtM. However this strategy fails to perform well in conditions of low transmission range where clusters get formed due to groups of nodes, moving to some spatially remote region. Quite obviously, since the agent is going to be in only one of the clusters, the other clusters would have no agents at all in them although the members belonging to those clusters may be well connected amongst themselves. The above mentioned issues cause no serious concern in the case of a multi-agent system. The probability of all the agents being trapped within the same small cluster is remote.

The Agent Model
An agent consists of the following three components:

1. The agent identifier *id*
2. The agent program *P*
3. The agent briefcase *B* (containing state variables)

The agent briefcase contains a set of network *state variables* which act as the memory of *mobile agents*. Examples of such *state variables* may be *link affinity, recency value* etc. An agent is capable of sharing the contents of its *briefcase* with other agents and nodes. The state variables may be updated if necessary before the agents leave the node. Further discussions on the agent communication protocols are detailed in subsequent sections.

The Optimal Agent Population? Intuitively, increase in agent population might seem to be beneficial to improve convergence. However, quite contrary to our intuition we observed that convergence does not necessarily improve with increase in the number of agents. To understand why, we performed a set of simulation experiments in [21]. From the results we concluded that in increasing the number of agents beyond a certain fraction of the number of nodes, a greater number of agents tend to rush towards the same set of nodes. This results in a higher queuing delay for agents and thus the aggregate performance is no better than relatively fewer agents. It was observed in [21] that the convergence curve saturates when the number of agents is half the number of nodes in the network. Figure 1 depicts the variation of average connectivity convergence with increase in agent population. To ensure this fraction, each host in the network generates a random integer at the commencement of the network. In our implementation, a host spawns an agent only if the randomly generated integer is even.

Agent TtM? An important question could be the optimal value of TtM. To answer this we evaluate the congestion introduced in the system due to variation in TtM. Let us assume that agents would take t millisecond to physically migrate from one node to another. Let us assume that our bounded region of ad hoc operation is A sq.mt., our transmission range R, the agent population P and the Time to migrate T msec. In an

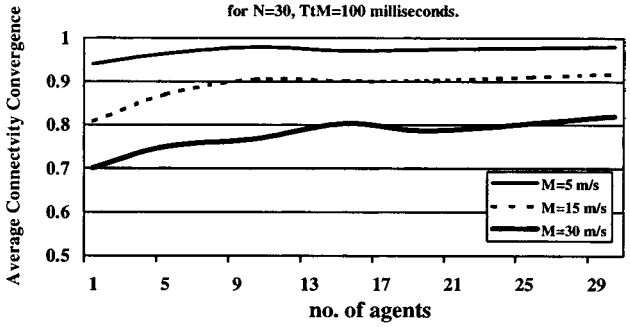

Figure 1. Average Connectivity Convergence against increasing agent population. Graph shows that greater number of agents does not imply better convergence.

average case where the topology is evenly distributed over the region A, the number of areas in which agents could migrate between nodes simultaneously, without mutual interference, equals A / ($\pi \times R^2$). Now since the nodes are distributed, the agents would also be found equally distributed in each of these areas in an average case. Thus in any area the number of agents would equal P_a where, $P_a = P \times (\pi \times R^2)/A$.

As each agent migrates at a time gap of T milliseconds and takes only t millisecond to do so, the medium will be free from agent traffic $[(T - t \times P_a) \times 100 / T]$ *percent* of the time. For example, if the bounded region of operation is 1500 × 1000 sq. m. and R is 250 m, and P and T are 15 and 100 milliseconds respectively for a 30 node network and t = 3 msec, then $P_a = 5.89$ approximately and the medium would be free from agent traffic during 94 percent of the time.

It would be interesting to point out that the value of TtM could be adaptively regulated depending on the state of the network. A system, which is highly mobile, can be dynamically tuned to a low TtM value so that the agents navigate faster meaning that stale topology information could be refreshed with haste. On the other hand, when data traffic is low, or network changes are not too frequent, the TtM could be raised to a suitably high value so that unnecessary power consumption in transmitting control packets could be reduced.

5.2. *Agent navigation: least-visited-neighbor-first algorithm*

The primary objective of agents is to deliver each node with information about other nodes in the system (including their link states). In order to achieve this with least overhead, we have designed a navigation strategy, which determines the path that agents must follow. Each agent moves on its own path, updates its briefcase with recent information gathered from other agents or nodes. Each node has a shared information cache (called a *blackboard*), on which agents can *overwrite* stale network-state values with more recent values. Hereafter we would use the word *blackboard* interchangeably with *information cache*. Nodes access this blackboard whenever they require knowledge about the network.

The efficacy of the agent navigation algorithm lies in its ability to make all the nodes in the network equally aware of the topology. Thus in an ideal scenario, a node which is located at the periphery of the ad hoc topology should be equally aware about the state of the network as any other node that might be located in the central regions of the topology. Our *least-visited-neighbor-first* algorithm, for controlling the navigation strategy of the agents, performs well close to the ideal expectations. An agent applies the least-visited-neighbor-first algorithm on the information cache of its host node (the host node is the node on which the agent currently resides) to decide on its next destination. The next destination is always a neighboring node of the host node.

On reaching a node N, an *agent program* P, performs the following steps:

1. Updates information cache of node N with any newer information available in its own *briefcase* B (we discuss the information exchange protocol in the following section)
2. Selects from the cache/*blackboard*, all the nodes that are neighbors of N.

3. Determines the node (among these selected neighbors) that has the least recency-token value. Observe that this is the least visited neighbor, as perceived by the node N at that instant of time (recall that the recency token value of a node is the number of agent visits that the node has received).
4. If this neighbor of N has not been visited in the previous 3 visits by other agents from node N, the agent selects this neighbor as its next destination. This *history* information about the last 3 visits are also found on the *blackboard*. In case the selected node has been visited in the recent past, the agent selects the second least-visited neighbor, and so on. This will ensure that multiple agents from the same host node do not choose the same destination consecutively.
5. After choosing the right destination, the agent updates its *next-destination node id* state variable with the chosen destination's node id and changes the *history* variables on the host node's blackboard with the *next-destination node id*.
6. Increments the host node's *recency token* value and stores this value against the host node's id in its own briefcase. The agent then resumes navigation.

Thus, if we consider a host node of an agent and its neighbors to be a sub-graph, then the agent always migrates to the node, which has had the least number of agent visits among the members of this sub-graph and has not been visited very recently. Since all the agents perform the same operation over the entire network, we can envision that agents attempt to visit all the nodes in the network with the same frequency. This is a typical characteristic of agent paradigms, where distributed decisions based on local states steer the system towards achieving a global goal.

Seeking a contradiction to the claim that agents achieve a global goal, let us assume that there is some node X, which has a very low recency value in comparison to all its neighbors. Now, every agent that visits the neighbors of this node X, would definitely choose node X as their next destination. This would continue as long as node X happens to be lesser in terms of its recency value. Thus soon node X would attain a recency value, which is greater than the least recency value of its neighbors. This means node X is no longer the least recency-valued node. It is quite interesting to realize that the difference in recency values would not be large in the first place because the agents always tend to visit the node that is falling back on the number of agent visits that it has received. However one exception to this is when a node gets isolated for sometime and then rejoins the network. We discuss this event in the following section.

5.3. Handling the event of agent oscillation between nodes

Let us consider a case where a node Z gets isolated from the topology and as a result does not receive agent visits for a long time. Now let us assume that it gets reconnected to the network after some time. Obviously, the recency token value of this node would be much less than the values of the others, which have continuously received agents at regular intervals. Now this isolated node on getting connected to the network would obviously have a rush of agents towards it. The agents would get serviced, go to some next destination, say D, that is a neighbor of Z. Now, since Z is obviously a neighbor of D, the agent would again come back to Z from D since node Z still happens to be the least in terms of the recency value. This continues until node Z is no longer the least

recency-valued-node in the neighborhood of node D. Thus we see that if a node gets isolated and then rejoins, the agents would oscillate over it and its neighbors, until its recency value surpasses some other neighbor's recency value. Oscillation might cause other nodes in the network to starve of agent visits and is unnecessary since these same oscillating agents hardly bring in new topology information.

In order to eliminate this oscillation, we have incorporated the following strategy. If a node finds that it does not receive agent visits for longer than a specific span of time, it resets its recency token value to zero. An agent that visits a node, finding that the node has a recency-token value of zero recognizes that the agent performed a *reset*. The agent performs the standard node-agent information exchange (*blackboard communication*) and then assigns the average of all its recency token values to this node's recency token. This prevents the oscillation of the same agent. However other agents may independently come to this node since they might still have a recency value for this node as the least value. This is desirable in order to expedite the process of information percolation towards a new node that was cut off from the network for sometime, and has rejoined it. Greater agent visits (i.e different agents visiting it) would catalyze this quick infiltration of information. Observe that by addressing the problem of agent-oscillation we implicitly take care of new nodes joining the network, since in the case of new nodes, the recency token values would also be zero.

One way of evaluating the performance of the agent navigation protocol would be to examine the celerity with which a topology change information gets propagated into the nodes of the system. We have defined a notion of *percolation* to test this issue. Percolation (P) has been defined as the rate at which, information regarding the entry of a new node, gets propagated to the existing nodes of the network. $P \times T$ is the percentage of nodes in the network that is aware of the new node entry after T milliseconds of the entry. We evaluate *percolation* in later sections for highly mobile scenarios, and show that our agent based mechanism performs well under it.

Unidirectional Links: We now discuss the impact of relaxing the assumption of bidirectional links in our protocol design. When nodes in a network have unequal transmission ranges, cases are possible where a node A is a neighbor of node B but node B is not a neighbor of node A. Such a situation may occur when B's transmission range is greater than A's. Clearly an agent visiting node A would not find B in A's *Neighbor List* and thus cannot hop to B. Thus the *affinity* field of link A-B would be marked infinity in the agents briefcase. An agent arriving at B, may find A in B's *Neighbor List*, and may choose to hop to A. The affinity of link B-A would not be infinity, since B can reach A in a single hop. Thus while link A-B and link B-A would exhibit identical affinity under the assumption of bidirectionality, assuming unidirectionality the two would be different. Since MARP does not require affinities of A-B and B-A to be equal, unequal transmission ranges at each node does not affect MARP.

6. Information exchange protocols in node-agent and agent-agent interaction

Infiltration of partial network information into the nodes is an asynchronous process, as the agents visit the nodes asynchronously. Thus it becomes acutely necessary to develop

strategies for information exchange (i.e. to accept only that information which is more recent than what the node / agent already possesses). It is a two-step process.

In step 1, the *recency-token values* of all the nodes stored in the information cache of the current host node is compared with the corresponding *recency-token values* of that node, carried in the briefcase of the agent. If the recency token of any node, say X, in the host node's information cache happens to be less than that in the agent's briefcase, then it is obvious that the agent is carrying more recent information about node X. In that case, the entire information about node X, in the host node's information cache (blackboard) is overwritten with information carried in the agent's briefcase. The information about node X that we refer to may be information regarding several attributes of the node, or even the network. The information could contain node X's link states with its neighbors, its remaining battery lifetime or even it's queue information. It is interesting to observe that the agents can now be reactively applied in several applications. For example, in the context of QoS routing, bandwidth reservation tables can be exchanged among a subset of nodes and even perhaps by only a subset of agents. However, in this paper we confine ourselves to distributing link state information only.

Step 1 is performed asynchronously by all agents as they arrive at their host nodes. This step helps the node to acquire all the recent information that it can gather from the agents. The agents however have not yet updated the state variables in their briefcases, and waits till the end of a TtM time duration. During this duration, if more agents arrive at the node, they perform step 1 immediately. Thus clearly, at any given time, the *blackboard* at a node contains more recent information than any individual agent resident at that node.

Step 2. When an agent is ready to migrate (i.e. after a waiting time defined earlier as TtM), step 2 is performed. In step 2, the agent copies the entire content of the *blackboard* into its own briefcase's state-variables. With the updated briefcase, the agent selects its destination on the basis of the navigation algorithm described previously. The agent then enters the agent queue for priority transmission. The blackboard at the host node is shown in Figure 2. The briefcase contains variables (data structures) that are an exact replica of the blackboard (except the history information).

6.1. Information aging: a predictive method for topology awareness

The foremost characteristic of a dynamic environment is that information is never absolute. Information collected by an agent or node regarding the link states of the network is constantly aging due to the mobility. However, the agents are pro-actively replenishing the cache of each node with newer or more recent information. Naturally, a node would be better topology aware, if the agent visit frequency could be increased upon it. The vice versa is true as well. In other words we could say that each time an agent visits a node, the node gets a snapshot of the network topology which is not accurate but less inaccurate than the previous snapshot it possessed. (It must be observed that a new snapshot might not have new information about all the nodes). Now, between any two consecutive snapshots, the relation is mobility i.e. if snapshot 2 is taken T seconds after snapshot 1, then snapshot 2 could be predicted (or derived) from snapshot 1 by displacing each of the nodes in snapshot 1 by a distance of $velocity \times T$, in the direction of motion of that node. Clearly, more the number of snapshots a node gets in a unit time, lesser is the information aging i.e. greater topology awareness. Thus to maintain optimal topology

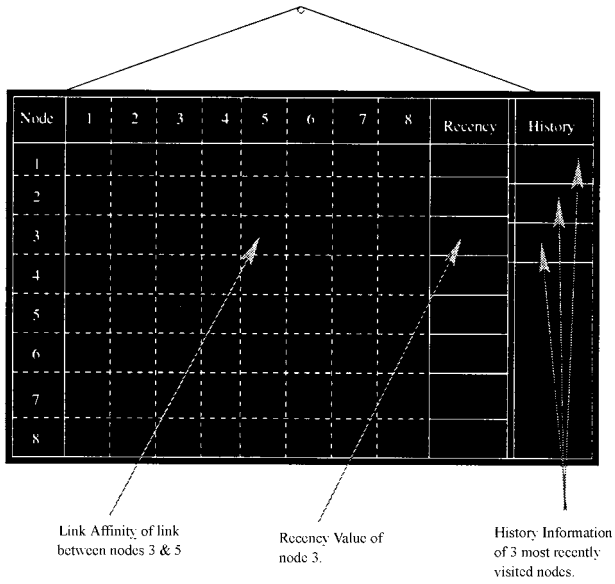

Figure 2. Diagrammatic representation of the *blackboard* at each node. Agents erase stale entries on the blackboard and overwrite them with more recent values. Nodes use the information on the blackboard for routing decisions.

awareness in nodes, the prediction mechanism comes handy in between agent visits, when local node information actually ages.

In our protocol, it is not possible to know the absolute speed or direction of motion of any node. Each node is only partially aware of the affinity of links in the network. Using a pessimistic approach, we incorporate a mechanism of reducing the affinity values of each link with the passage of time. Thus at every interval of t time units, the affinity values of all the links is reduced by *velocity* $\times$ *t* time units on the *blackboard* at each node. This is a conservative approach that assumes that *link affinity* is always decreasing. Thus cases are likely that a network link exists in reality, but a node has eliminated it from its blackboard. However, the vice versa is extremely unlikely, indicating that a node is never misled to believe that a link exists although in reality it does not.

7. Simulation set-up

We have used our own simulator (as shown in Figure 3) for evaluating the performance of MARP. The simulation region is a bounded area of 1500 $\times$ 1000 sqm. Nodes are initially placed randomly over this region. The mobility model assumed is random waypoint - each node moves linearly towards their chosen destination, on reaching their destinations, each node optionally waits for a random span of time, chooses a new destination and starts moving towards this newly chosen destination. To observe the impact of mobility we have, in certain cases, assigned a predefined uniform velocity to all the nodes. The transmission range for all communication has been assumed to be constant. We do not use agent loss due to erroneous transmission. We assume that the *agents* would be transmitted reliably by

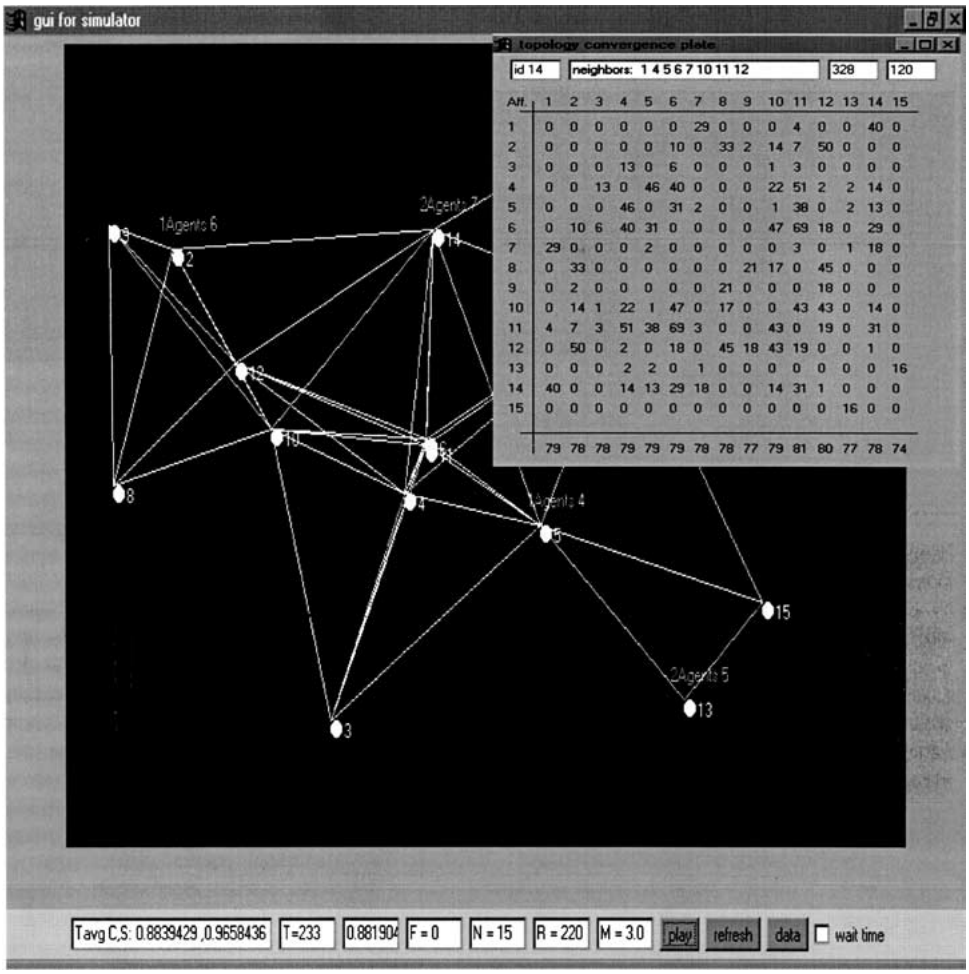

Figure 3. The simulator environment snapshot with N = 15 and M = 30m/sec: the nodes move about with agents migrating between nodes. The inset window pops up if a mouse is clicked on any of the nodes. The window shows the *blackboard* at node 14 (history values have not been shown). The affinity values are in hundreds of milliseconds. The row shown below this affinity matrix displays the recency token values of all the nodes in the network. A good navigation strategy ensures that these recency token values exhibit small standard deviation, implying that the topology is explored homogeneously. The other parameters for the simulation are shown below in the text boxes. The standard deviation of the recency values calculated is = 1.52606, mean = 78.266.

the MAC layer. However, if the MAC layer fails to transmit the agent, it must notify the upper layers before dropping the agent. The upper layer may spawn a new agent later, to maintain optimal agent population.

The multi-agent framework is operational over this infrastructural backbone of mobile nodes. During commencement, each node generates a random number locally. If the generated number happens to be even, the node spawns an agent, otherwise it does not.

This indicates that the number of agents that appear within the network infrastructure is approximately half the number of nodes in the network. The agents then jump asynchronously from one node to another at an interval of TtM milliseconds, carrying information that it gathers from its host. Now, although the agent-jump is a unicast, it has to be broadcast in the wireless environment. Thus all one-hop neighbors of the host node can now hear the agent. On hearing the agent, they know the existence of the host node as its neighbor (please note that the agent carries the ID of both its present host node and its destination node). The destination node accepts the agent and the others discard the agent packet after extracting the neighbor information from it. An advantage of using this protocol is that explicit neighbor discovery at the MAC layer can now be done away with. The agents while on their topology traversal, implicitly perform neighbor discovery. On reaching the next node, the agents exchange information with their new host, wait for TtM milliseconds and then again jump to a newly selected node. Thus nodes that were totally unaware of the network topology at the point of commencement, begin to gather network information through the agents. Since the multi-agent framework is a pro-active scheme, the nodes in the network are always kept updated with partial (or approximate) topology information. A modified link state algorithm executed locally over this partial information yields the most stable route through which data communication is initiated.

8. Supporting data communication

We are now in a position in which each of the nodes has a local view of the network topology i.e., each node is topology aware. Also, the mobile multi-agent framework is proactively replenishing the information cache of each node with fresh topology updates. This leads to a scenario where conventional route discovery is no longer necessary. More explicitly, nodes can now determine the best route locally and initiate the sending of data packets through it. After a point of time, if the caller node finds that the chosen route has attained a low stability (indicating that a route error is imminent), the node computes a new, better stable route from the local *blackboard* and redirects data packets through the later. This adaptive route selection facilitates continuous communication through multiple paths in the temporal domain. *Thus we can envision that as long as two nodes remain connected, they will always be able to get at least one route through which communication can continue*. In the case of multi-route availability, the best route can always be selected. Quite perceivably, the adaptive selection of best routes guarantees an uninterrupted communication session between two nodes irrespective of node mobility. Besides, packet drops due to route breakage, and latency due to re-routing, has been completely eliminated.

Due to our conservative approach in link state prediction, we find that the probability of route errors almost becomes negligible. In other words, a situation can hardly occur when a route error takes place without the sender not anticipating it. Note that even if nodes move non-linearly (in curves and turns), our protocol performance remains unaffected. This is because we conservatively assume in our prediction mechanism, that two nodes are always moving away from each other with relative velocity M. However, due to this conservative prediction, a sender node may not initiate communication because it might believe that a link does not exist while in reality it does. Our simulation experiments have

yielded expected results. In the worst case however, if a route error occurs, the caller node need not initiate a route discovery all over again but just needs to locally determine the second best route available, and resume data transfer through it. This obviously reduces control packet generation to a nil and minimizes end-to-end delay substantially.

9. Agent vs non agent systems: a comparative study

We discussed that one of the major problems in distributed mobile systems (as MANETs), is the tracking of mobile hosts while keeping congestion overheads low. Route discovery or path maintenance incurs huge overhead due to control packet flooding in reactive mechanisms. This problem becomes acute when the network is considerably loaded. More precisely, the total number of control and data packets in the output queues of all the nodes in the network at any instant of time, is a measure of the load on the network. During a route discovery and subsequent data transmission from a source to a destination, this number increases rapidly with time and gradually decays down. The increase in the number of control packets in the host queues is exponential with the increase in the number of neighbors. This in turn causes greater queuing delay for the subsequent data packets. Over and above, appearance of control packets in the network also depends on the number of communications started by caller nodes over a span of time (indicating that the network gets flooded with route requests as many times). As an outcome of so many dependencies, end to end delays or available bandwidth of the system almost become unpredictable. This might in turn affect the survivability of the system if the network does not get time to absorb the packets accumulated in the output queues.

Proactive protocols in literature have generated unnecessary control packets even while the traffic and / or mobility is low. This has been shown to have dire effects in the sense that battery power of nodes gets consumed unnecessarily.

In contrast our agent-based protocol exploits the merits of both reactive and proactive protocols. Agents are not flooded in the network indicating that the congestion does not increase with the number of hops. In fact, as discussed in previous sections, the agent traffic always remains constant for a particular value of TtM. This enables a better estimation of the end to end delay of the system. On the other hand, when agents jump from one node to its neighbor, the other nodes in the same one hop neighborhood listen to the agents to extract neighborhood information. This minimizes the overhead of performing neighbor discovery separately. In addition, to reduce unnecessary agent traffic during low load and / or low mobility conditions, the agent TtM could be adaptively regulated to conserve power at individual nodes. Together, it seems that the agent system, besides being more survivable even under stringent conditions of high traffic and high mobility, is adaptable to the requirements of the network.

The multi-agent system we discuss in this paper could be implemented using a message passing model. One way of doing this may be to replicate the agent program (P) at all the nodes, and propagate the agent's *briefcase B* as a message/packet. Although correct, such a strategy may lack modularity of implementation. Also, using agents, the system could be much more flexible, allowing agent subsets to perform different set of tasks, autonomously. For example, if a few nodes are high priority nodes in a network, a set of agents may be programmed such that they visit these nodes more often.

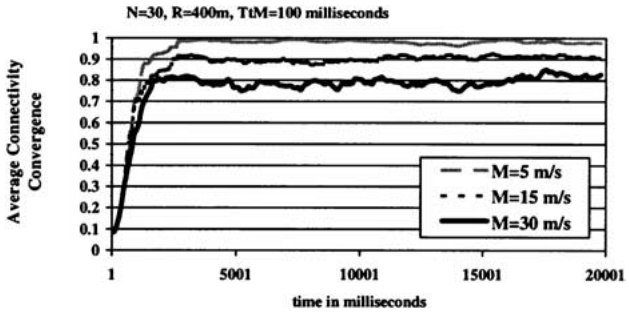

Figure 4. Variation of connectivity convergence with time for different mobility with TtM = 100 msec.

9.1. Protocol performance

We have presented the performance analysis of our simulation in this section. We initially present the performance of the agent based system without the predictive mechanism. We then present the simulation results as observed after incorporating the predictive mechanism.

Figure 4 shows the average connectivity convergence of an ad hoc network for 30 nodes, with 15 agents exploring the network. The agents are allowed to migrate from a node at intervals of 100 milliseconds, i.e., TtM = 100 msec. The transmission range has been assigned as 400 m. This transmission range has been found to give us a more or less connected network. In these cases we see that mobility plays an important role. At higher mobility the rate at which link information get stale is proportionally high. This makes it more challenging for the agent framework since the system now demands that they deliver link make-break information across the topology at a faster rate.

As per expectations, we observe that performance degrades at very high mobility. To compensate for increase in mobility, the solution would be to decrease agent TtM. It is evident from the graph in Figure 5 that for mobility 30 m/s the performance is significantly different for high and low TtM. Thus the agents could be adaptive to the average mobility of the nodes and tune its TtM accordingly. Quite obviously, by decreasing agent TtM, we can achieve better convergence. However, a low TtM would imply that nodes get more agents per unit time and thus the network congestion due to agent traffic would also

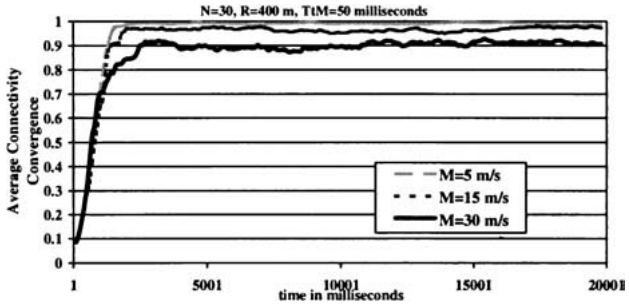

Figure 5. Variation of connectivity convergence with time for different mobility with TtM = 50 msec.

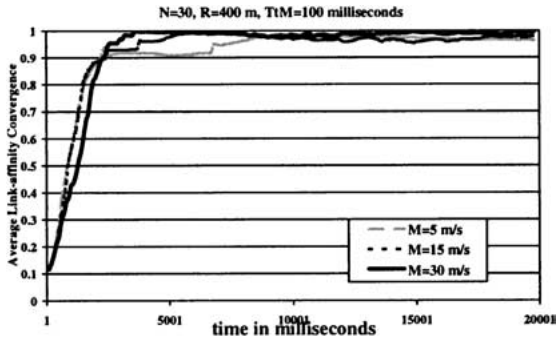

Figure 6. Variation of Average Link Affinity Convergence with time for different mobility with TtM = 100 msec.

increase. A predictive mechanism on network topology, discussed in Section VII A, can achieve better result without lowering TtM.

We have analyzed the performance of our prediction mechanism and the results are presented in Figure 6 and 7. This mechanism ensures that the node is never misled to believe that a path exists for successful message transfer when the path might actually fail before data transmission is complete. The Link-Affinity convergence graphs show satisfying results as the curve always remains above 98 percent once the topology information has stabilized. Put differently, the probability that link-affinity information of a path is acceptable, is greater than 0.98 on an average. As a result, if a node believes that a path exists in the network, the probability that it does not, is only 0.02. These figures hold good for agent TtM = 100 msec.

Figure 8 shows the sensitivity of Link-affinity convergence to mobility. The results shown are for MARP, when used with the predictive mechanism. Clearly, MARP performs well when the velocity of nodes is below 45 m/s. At greater velocities, the performance with TtM = 100 msec degrades. The performance with TtM = 50 msec remains above 0.92 even at 45 m/s. With TtM = 50 msec, convergence deteriorates at velocities above 60 m/s. Since many ad hoc applications are limited to much lower node velocities, MARP may be considered suitable for mobile ad hoc networks.

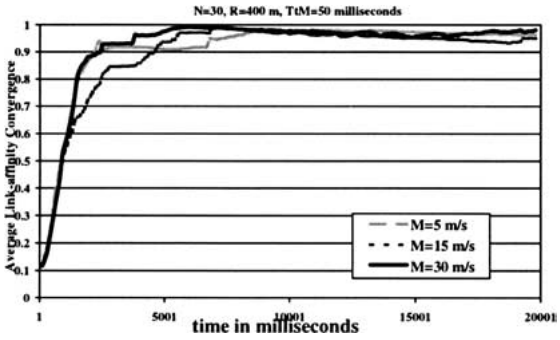

Figure 7. Variation of Average Link Affinity Convergence with time for different mobility with TtM = 50 msec.

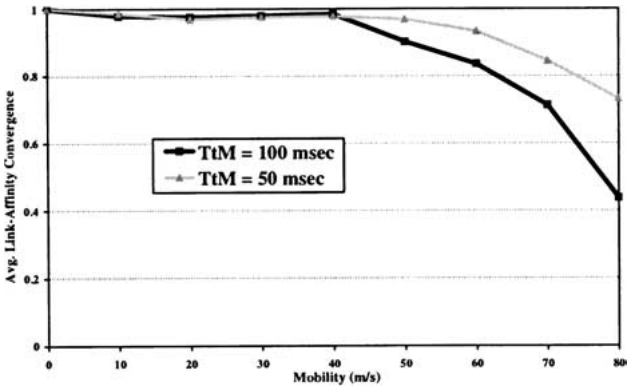

Figure 8. Variation of Average Link-Affinity convergence with node mobility.

The performance of MARP has been evaluated for mobile topologies with a large diameter. Observe that a graph with a larger diameter would require agents to traverse greater number of hops (on average) to maintain comparable topology awareness. To observe the performance of MARP in large-diameter networks, we reduced the transmission range to 200 meters and increased the number of nodes in the bounded region to 60. This maintains connectivity for a network that is confined to a bounded region of 1500×1000 sq. meter. Figure 9 shows the results for the simulation. The TtM used was 100 msec. As evident from the graph, average Link-Affinity convergence is only marginally worse in comparison to previous scenarios, although the diameter of the graph is now higher. With a lower TtM, MARP would exhibit better performance than shown in the Figure 9. This suggests that MARP is intrinsically scalable to network size and mobility. Also, with increase in number of nodes, the number of agents in the system increases. Increase in agent population increases the celerity of topology information distribution, almost exponentially. Thus even when the node density is high, performance of MARP exhibits stability.

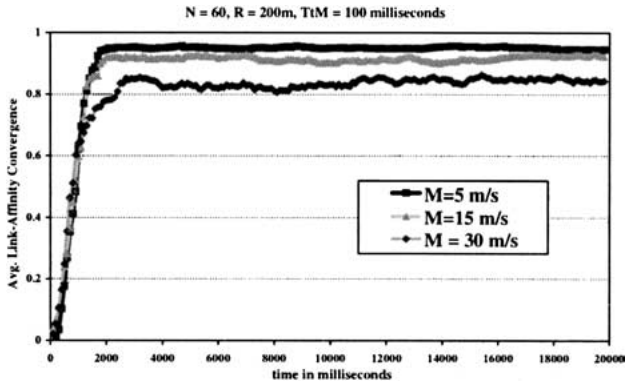

Figure 9. Average Link convergence for network with large diameter.

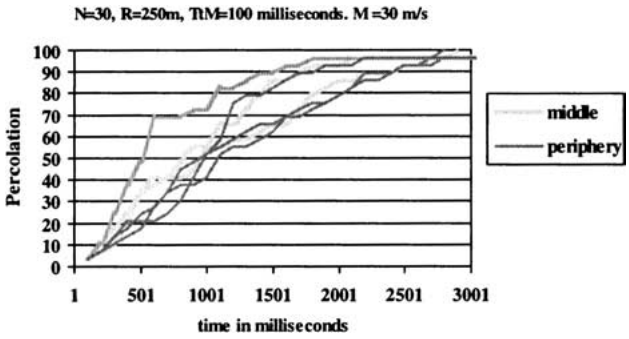

Figure 10. Percolation of node information of time.

Convergence is a metric that reflects the efficiency of the agent framework on the basis of its ability to homogeneously spread topology information. We had also defined percolation as another metric that captures the rate at which information propagates in the network. To capture this metric we have thrown in new nodes in the network topology while the simulation is running and we have examined the rate at which the other nodes in the system learn about this new entry. A point that requires mention is that, contrary to intuitive belief that a node thrown in the center of the topology would have its information propagated much faster, than if the same is thrown towards the periphery, we find that our agent system carries the new information to other nodes with almost equal celerity. Thus the curve in the Figure 10 accentuates on the efficiency of the agent navigation protocol. From this graph we find that the new node entry gets propagated to all the other nodes within a span of approximately 3 seconds and the difference between peripheral appearance and central appearance is small.

10. Discussion

The performance evaluation of the agent-based mechanism is encouraging, even in highly mobile multi-hop environments. In addition, the capability of the agent framework to adapt its TtM to the load and/or low mobility scenarios further regulates on the control overhead generated due to agent traffic. Percolation is a pointer to how the agent system accommodates new nodes into the system quite seamlessly. The scalability issues get implicitly handled as new nodes spawn agents with one half probability (indicating that the agent-node ratio is maintained). And of course, most importantly, the impact of topology awareness obviating route discovery (and rediscovery on route error) is perhaps the most significant gain in terms of the survivability of the system.

Moreover, we believe that the capacity of the agent-based protocol is just not confined to the listed set of available features. We suggest that the agent system can be extended to carry/fetch specific information on demand. For example, queue, reservation table information, battery power levels, error information etc. may be transferred through wondering agents; or even through only a specific subset of agents to reduce on the bandwidth consumption. As a whole the notion is to optimally utilize a traveling entity in the network to carry out multi-purpose information transfer. The agent seems to be a suitable idea.

11. Conclusion

This paper proposes a multi-agent framework capable of making nodes in a mobile ad hoc network, topology-aware. Topology awareness in this paper, implies the ability of a node to maintain the link states of a network, with sufficient accuracy. Although confined to link states, our multi-agent framework may be viewed as a overlaid backbone, capable of propagating other relevant information, that different layers may expect from the network. Once aware of the network links, nodes are capable of streaming packets through routes, calculated locally from gathered information. We show how the agent system keeps control overhead low and maintains link-state convergence, even at high mobility. We have not considered agent loss in this paper. We have assumed that MAC level unicasts are reliable. Situations are also possible in which agents may need to be destroyed in order to maintain the optimal node-agent ratio. We have not addressed such problems of regulating agent population, since they are independent problems by themselves. However, under the adopted set of assumptions, simulation results are encouraging, indicating that a proactive agent based routing protocol like MARP is efficient and effective across a large cross section of scenarios.

References

1. V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview," *IEEE Communication Magazine*, July 1998.
2. R. Gray, D. Kotz, S. Nog, and G. Cybenko, "Mobile agents for mobile computing," Technical Report PCS-TR96-285, Department of Computer Science, Dartmouth College, Hanover, NH 03755, May 1996.
3. D. B. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in T. Imielinski and H. Korth, (eds.), *Mobile Computing*, Kluwer Academic Publ., 1996.
4. Z. J. Haas, Milcom'97 Panel on Ad-hoc Networks, http://www.ee.cornell.edu/haas/milcom-panel.html
5. S. Corson, J. Macker, and S. Batsell, "Architectural considerations for mobile mesh networking," *Internet Draft RFC Version 2*, May 1996.
6. S. Das, Somprakash Bandyopadhyay, and K. Paul, "An adaptive framework for QoS routing via multiple paths in ad hoc wireless networks," *Proc. of GLOBECOM*, 1999.
7. C.-K. Toh, "A novel distributed routing protocol to support ad-hoc mobile computing," *IEEE International Phoenix Conference on Computer and Communications (IPCCC'96)*.
8. E. M. Royer and C. K.Toh, "A review of current routing protocols for Ad-Hoc Mobile Wireless Networks," *IEEE Personal Communications*, April 1999.
9. C. Perkins and E. Royer, "Ad hoc on-demand distance vector routing," *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90–100. Online. Available. http://beta.ece.ucsb.edu/eroyer/aodv.html
10. K. Paul, S. Bandyopadhyay, D. Saha, and A. Mukherjee, "Communication-Aware Mobile Hosts in Ad-hoc Wireless Network," *Proc. of the IEEE International Conference on Personal Wireless Communication*, Jaipur, India, Feb. 1999.
11. Y.-B. Ko and N. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," *Proceedings MOBICOM 1998*, Dallas, Tx.
12. S. Appleby and S. Steward, "Mobile software agents for control in Telecommunications networks," *BT Technology Journal*, vol. 12, no. 2, pp. 104–113, April 1994.
13. A. Bieszczad, B. Pagurek, and T. White, "Mobile Agents for Network Management," *IEEE Communications Survey*, July 1998.
14. T. Magedanz, K. Rothermel, and S. Krause, "Intelligent agents: An emerging technology for next generation telecommunications?" *Proc. INFOCOM'96*, San Francisco, CA, 1996.
15. M. Baldi, S. Gai, and G. P. Picco, "Exploiting code mobility in decentralized and flexible network

management," in K. Rothermel and R. Popescu-Zeletin, (eds.), *Mobile Agents, Lecture Notes in Comp. Sci. Series*, vol. 1219, pp. 13–26, Springer, 1997.
16. S. Krause and T. Magedanz, "Mobile service agents enabling intelligence on demand in telecommunications," *Proc. IEEE GLOBCOM '96*, 1996.
17. R. Schoonderwoerd et al. "Ant-based load balancing in telecommunications networks," *Adaptive Behavior*, vol. 5, no. 2, p. 169207, 1997.
18. G. Di Caro and M. Dorigo, "Mobile agents for adaptive routing," in *Proceedings of the 31st Hawaii International Conference on Systems*, January 1988.
19. N. Minar, K.H. Kramer, and P. Maes, "Cooperating mobile agents for dynamic network routing," in Alex Hayzeldon, (ed.), *Software Agents for Future Comunications Systems*, chapter 12, Springer-Verlag, 1999.
20. S. Bandyopadhyay and K. Paul, "Evaluating the performance of mobile agent based message communication among mobile hosts in Large Ad-Hoc Wireless Networks," The Second ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems, in *Conjunction with MOBICOM 99*, Washington, USA, August 15–19, 1999
21. J. Dale, "A Mobile Agent Architecture for Distributed Information Management," Thesis submitted for the degree of Doctor of Philosophy. University of Southampton, Department of Electronics and Computer Science, September 1997.
22. R.R. Choudhury, S. Bandyopadhyay, and K. Paul, "A distributed mechanism for topology discovery in ad hoc wireless networks using mobile agents," Proc. of the First International Workshop on Ad Hoc Networks, ACM/IEEE MobiHoc 2000, in *Conjunction with MobiCom 2000*, Boston, USA.