TOPOLOGY DISCOVERY IN AD HOC WIRELESS NETWORKS USING MOBILE AGENTS

Romit RoyChoudhuri[1], S. Bandyopadhyay[2], Krishna Paul[3]

[1] Department of Computer Sc and Engg
Haldia Institute of Technology
Haldia, West Bengal, India

[2] PricewaterhouseCoopers, Saltlake Technology Center
Sector V, Calcutta 700 091, India
{somprakash.bandyopadhyay@in.pwcglobal.com}

[3] Cognizant Technology Solutions, Sector V, Saltlake
Calcutta 700 091 India
{PKrishna2@cal.cts-corp.com}

Abstract. Extensive research on mobile agents has been rife with the growing interests in network computing. In this paper, we have discussed a mobile multi-agent-based framework to address the aspect of topology discovery in ad hoc wireless network environment. In other words, we have designed a multi-agent based protocol to make the nodes in the network *topology aware*. Our primary aim is to collect all topology-related information from each node in the network and distribute them periodically (as updates) to other nodes through mobile agents. The notion of *stigmergic communication* has been used through the implementation of a shared information cache in each node. Moreover, we have defined a concept of *link stability* and *information aging* based on which a predictive algorithm running on each node can predict the current network topology based on the current network information stored at that node. We have demonstrated through performance evaluation of a simulated system that the use of mobile multi-agent framework would be able to make each node in the network *topology aware*, without consuming large portion of network capacity. As a direct outcome of infiltrating topology information into the nodes, the foundations for designing efficient routing scheme, distributed network management and implementing communication awareness get automatically laid.

1. Introduction

A mobile agent is a program that can move through a network under its own control, capable of navigating through the underlying network and performing various tasks at each node independently [1]. Mobile agents are an effective paradigm for distributed applications, and are particularly attractive in a dynamic network environment involving partially connected computing elements [2]. In this paper, we propose to use a mobile multi-agent-based framework to address the aspect of topology discovery in a highly dynamic ad hoc wireless network environment [3,4,5,6]. In other words, we have designed a multi-agent based protocol to make the nodes in the network **topology aware**. As a direct outcome of infiltrating topology information into each node, the foundations for designing efficient routing scheme, distributed network management and implementing communication awareness [7] get automatically laid.

As an example, 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 between any source and destination[5]. The conventional proactive routing protocols that require to know the topology of the entire network is not suitable in such a highly dynamic environment, since the topology update information needs to be propagated frequently throughout the network. On the other hand, a demand-based, reactive route discovery procedure generates large volume of control traffic and the actual data transmission is delayed until the route is determined. Because of this long delay and excessive control traffic, pure reactive routing protocols may not be applicable to real-time communication. At the same time, pure proactive schemes are likewise not appropriate for the ad-hoc network environment, as they continuously use a large portion of the network capacity to keep the routing information current [4]. In this work, we have demonstrated through performance evaluation of a simulated system that the use of mobile multi-agent framework would be able to make each node in the network topology aware without consuming large portion of network capacity. This would eventually help us to implement a proactive routing protocol without much overhead.

Mobile agents or messengers that hop around in the network are a novel solution to the problem of topology discovery. The agents hop from node to node, collect information from these nodes, meet other agents in their journey, interact with both to collect updates of parts of the network that they have not visited or have visited a long time back**,** and gift these collected data sets to newly visited nodes and agents. A node therefore receives updated information about the network from the agents visiting them at short regular intervals. Initially when the network commences, all the nodes are just aware of their own neighbors and are in a *don't-know-state* regarding the other nodes in the system. However with agent navigation beginning, the nodes slowly get information about the other nodes and their neighbors.

For example, let us assume that an agent migrates at every K time tick between nodes. At time t_0, each of the nodes has only information about their immediate neighbors. At time = t_0 + K, an agent jumps to a node with the information that it has about its previous host node. Thus the current host node now has data about a new neighboring node and its neighbors (since the agent has carried this information to it). In the next K time tick, a node gets information regarding two more nodes from another agent. It is to be noted that by controlling K, it is possible to control the agent traffic in the network. Moreover, the agent would always migrate from a node to only one of its neighbor after K time-tick. So, the network would never get flooded with propagation of agents.

2. Related Work

Currently, there is a growing interest in using mobile agents as part of the solution to implement more flexible and decentralized network architecture [1, 8]. Most research examples of the mobile agent paradigm as reported in the current literatures have two general goals: reduction of network traffic and asynchronous interaction. Some authors have suggested that agents can be used to implement network management [9, 10] and to deliver network services [11].

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]. Of particular interest is a technique for indirect inter-agent communication, called stigmergy, in which agents leave behind the information in the cache of the node that they have visited. Stigmergy serves as a robust mechanism for information sharing. Worker ants that leave pheromone trails when they venture outside their nest are using stigmergic communication. This notion has been used in [13,14]. Mobile agents are on the use for multifarious purposes ranging from adaptive routing [13], distributing topology information [14], offline message transfer[15] and distributed information management [16].

In this paper, our primary aim is to collect all topology-related information from each node in ad hoc wireless network and distribute them periodically (as updates) to other nodes through mobile agents. The notion of stigmergic communication has been used through the implementation of a shared information cache in each node. The basic idea of using agents for topology discovery has been explored in MIT Media Lab [14] earlier with certain limitations : first, node mobility and its effect on system performance has not been quantified. Second, the information convergence (convergence of the difference between actual topology information and the topology information as perceived by a node at any point of time) and its relationship with number of agents and agent migration frequency has not been clearly defined. Third, the navigation strategies used do not ensure a balanced distribution of recent topology information among all the nodes. We have tried to overcome these difficulties. Moreover, we have defined a concept of link stability and information aging based on which a predictive algorithm running on each node can predict the current network topology based on the current network information stored at that node.

3. Description of Relevant Terms

3.1 Link-Affinity

In a wireless environment, each node n has a wireless transmitter range. We define the neighbors of n as the set of nodes within the transmission range R of n. It is assumed that when node n transmits a packet, it is broadcast to all of its neighbors. However, in the wireless environment, the strength of connections to all the members of the neighbor set with respect to any node n are not uniform. For example, a node m in the periphery of the transmission range of n is weakly connected to n compared to a node u which is closer to n. Thus, the chance of m going out of the transmission range of n due to an outward mobility of either m or n is more than that of u.

Link-Affinity α_{nm}, associated with a link between two nodes n and m, is a prediction about the span of life of that link in a particular context [7]. For simplicity, we assume α_{nm} to be equal to α_{mn} and the transmission range R for all the nodes are equal. To find out the affinity α_{nm}, node n sends a periodic beacon and node m samples the strength of signals received from node n periodically. Since the signal strength perceived by m is a function of R and the current distance r between n and m, we can predict the current distance r at time t between n and m. If M is the average velocity of the nodes, the worst-case average affinity α_{nm} at time t is (R - r)/M, assuming that at time t, the node m has started moving outwards with an average velocity M. If m and n are not neighbors of each other, $\alpha_{nm} = 0$.

3.2 Recency

A major aspect underlying the infiltration of topology information into mobile nodes is that the information carried must be recognized with a degree of correctness. Since the agent navigation is asynchronous and there is an obvious time gap between the procurement of information by an agent from one node and its delivery by the same agent to another node, it becomes imperative to introduce a concept of *recency of information*. For example, let us assume two agents A_1 and A_2 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 A_1 or agent A_2 ?

To implement that, every node in the network has a counter that is initialized to 0. When an agent leaves a node after completing all its tasks at the node, it increments that counter by one. We term this counter as *recency token*. An agent while leaving a node (after completion of all the necessary information exchange and calculations) stores the new value of the incremented recency token against the node's ID within its data structures and continues navigating. Thus at any point of time, 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 x, then the agent carrying the higher recency token value of node x has more current information about it. The far-reaching advantages derived from this scheme would be pointed out in further illustrations where the recency token is extensively used.

3.3 Time-to-Migrate (TtM)

An agent visiting a node is not allowed to migrate immediately to another node. In other words, 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 1 msec. to physically migrate from one node to another. So, in this example, the wireless medium would be free from agent traffic 99% of the time.

3.4 Average Connectivity Convergence

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

Let l^a_{nm} be the link status (0 for disconnectivity and 1 for connectivity) between node n and m as perceived by node a at any instant of time and l_{nm} is the actual link status between node m and n at that instant of time. Information about link status l^a_{mn} is said to converge at node a iff $l^a_{mn} = l_{mn}$. Thus, connectivity convergence of link between n and m at node a, $\gamma^a_{nm} = 1$, if $l^a_{nm} = l_{mn}$ and 0 otherwise. *Connectivity convergence* of node a for all links in the network, γ^a is defined as: $\gamma^a = (\Sigma_{\text{for all node-pairs i-j}} (\gamma^a_{ij}))$ / total number of node-pairs

At some instant of time, if $\gamma^a = 1.0$, it implies that the topology information at node a is exactly the same as the actual network topology at that instant of 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 matches at any instant of time with the actual link-status, then $\gamma^a = 44/45 = 0.98$.

The *average connectivity convergence*, $\gamma_{avg} = (\Sigma_{\text{for all nodes k}} (\gamma^k))$ / number of nodes

3.5 Average Link-affinity Convergence

Average connectivity convergence quantifies the deviation of actual network topology with the network topology perceived by individual nodes in 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} is the actual affinity between node m and n at that instant of time. Information about link status α^a_{mn} is said to converge at node a iff $\alpha^a_{mn} <= \alpha_{mn}$. As indicated earlier, affinity is a worst-case prediction about the span of life 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 will accept the perception of node a about the link-affinity between n and m. However, if $\alpha^a_{mn} > \alpha_{mn}$, we will consider this as an overestimate by node a about the link affinity between n and m and we reject the perception.

Thus, *link-affinity convergence* of link between n and m at node a, $\lambda^a_{nm} = 1$, if $\alpha^a_{mn} <= \alpha_{mn}$ and 0 otherwise.

Link-affinity convergence of node a, λ^a, for all links in the network, is defined as:
$\lambda^a = (\Sigma_{\text{for all node-pairs i-j}} (\lambda^a_{ij}))$ / total number of node-pairs At some instant of time, if $\lambda^a = 1.0$, it implies that the topology information at node a is 100% acceptable, so far as affinity-based prediction mechanism is concerned.

The *average link-affinity convergence*, $\lambda_{avg} = (\Sigma_{\text{for all nodes k}} (\lambda^k))$ / number of nodes

4. Issues in Implementing Agent Paradigm

4.1 Single vs. Multiple Agents

The topology traversing could well be performed using a single agent. However this strategy fails to perform well in conditions of low transmission range where clusters gets formed due to groups of nodes, moving to some spatially remote portion of the bounded region. Quite obviously, since the agent is going to be in only one of the clusters, the other clusters have no agents at all in them although the members belonging to those clusters may be connected to quite a number of other nodes. Thus, the deprived clusters can no more get topology maps and might even get misled by the old, and thus incorrect, information they might be having regarding the connectivity of the network.

Although we have not addressed the issue of agent loss (during transit) in this paper, it can be well understood that the single agent system would also suffer in the event of the agent getting lost during transit. This consequently means that the system no longer has an agent and the nodes are not even aware of it, since each would indefinitely wait for an agent to come to it.

The above mentioned issues cause no serious concern in the case of a multi-agent system. There are two factors to be considered in multi-agent system :
The queuing factor: With the increase in number of agents, less and less number of nodes is free of agents at any instant of time. In other words, the average number of agents in the nodes' agent queues increase. Since more and more agents are held in agent queues, the effective number of agents in the system decreases. This decrease in the effective number of agents compensates for the advantage of having more agents in the system which otherwise means that the convergence could be better.
The issue of common intentions in multi-agent systems: As the number of agents increase, the information that each of them get after the exchanges become identical and thus the corresponding decisions that they take regarding migration gradually becomes similar [14]. As a result agents crowd together at certain parts of the topology and the very essence of homogeneous topology exploration gets lost. Thus, this inhibits the prospective gain in increasing agent population and the increase in agent population also increases the traffic/congestion in the system.

4.2 Choosing the Optimal Agent Population

It might be of interest to observe that average connectivity convergence does not necessarily improve with the increase in number of agents in the system. It is this observation in our work that served as the motivation to analyze the dependence of performance on agent population and agent TtM. Thus we have determined the optimal number of agents in a system of N nodes to obtain the best optimal service.

Fig.1 show the analysis curves of average connectivity convergence against varying number of agents for different mobility and TtM = 100 ms.. It can be seen from the graph that the performance of the agency improves with greater number of agents up to a certain point after which the performance is not discernable. Increase in agent population in the system clearly indicates the increase in congestion in the system. Thus, to keep congestion under control, we rationally choose that agent population for which the agent traffic is tolerable and the requirements are well met. We have chosen the critical agent population to be *half the number of nodes* in the system.

4.3 Spawning of Agents

Each node at its start-up process generates a random number. We propose that, if the number is even, then the node spawns an agent in the network. This indicates that the network commences with an agent population which is roughly half the number of nodes in the network.

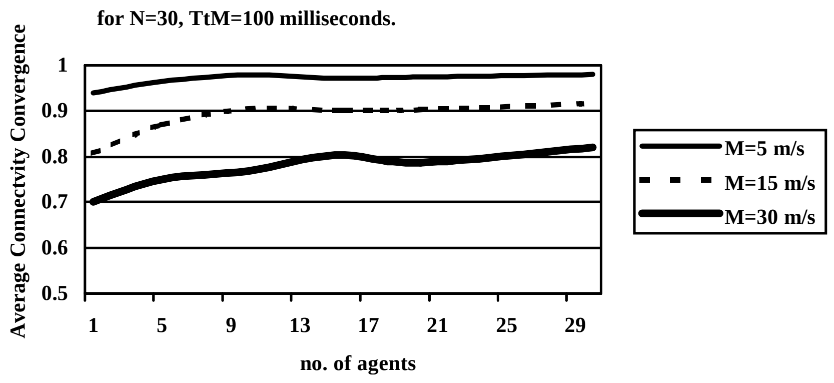


Fig.1. The variation of connectivity convergence against number of agents in the system, analyzed for different mobility values with TtM =100 ms

5. Topology Discovery Using Agents : Basic Mechanism

5.1 Navigation Algorithm for Optimal Throughput and Quick Convergence

The navigation algorithm must ensure that all member nodes of the network update themselves uniformly, irrespective of their position and state in the network. We have designed an algorithm, called *least-visited-neighbor-first* algorithm, and implemented it in our simulation model to estimate its effectiveness.

In this algorithm the agents select destination nodes that has the least recency_token value, implying that the node has been visited less frequently. An agent, residing in node n at any instant of time (termed as current host node of the agent), does the following:

1. Update information cache of node n with the information available in its information cache.
2. Selects all the nodes that are neighbors of n.
3. Evaluates the minimum value of the recency token of these selected neighbors from the information cache. This is the least visited neighbor, as perceived by the agent's host node n at that instant of time.
4. If this neighbor of n has not been visited in the last 3 visits by other agents from node n, select this neighbor as next destination. This history information is stored in the nodes. Else, select the second least-visited neighbors, and so on. This will ensure that multiple agents from same host node do not choose the same destination consecutively.
5. After choosing the right destination, it updates its next_detination node id with the destination node's id, changes the history table of the host node n with this newly selected node id.
6. Increments the host node's recency token value and stores this value against the host node's id within its own information cache.
7. The agent resumes navigation.

5.2 Handling the Event of Agent Oscillation between Nodes

Let us consider a case where a node gets isolated from other nodes in the network and as a result does not receive agent visits for a long time. Now let us assume that it gets connected to the network once again 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 and again come back to this newly connected node until the recency value of the newly connected node is no longer the least among all the recency tokens of the other nodes in the network. This means 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 from the perspective of percolating current 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 node performed a reset. The agent performs the standard node-agent information interchange 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 percolate a new node's information fast. Greater agent visits (i.e different agents visiting it) would catalyze this quick infiltration of information.

5.3 Information Exchange 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 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.

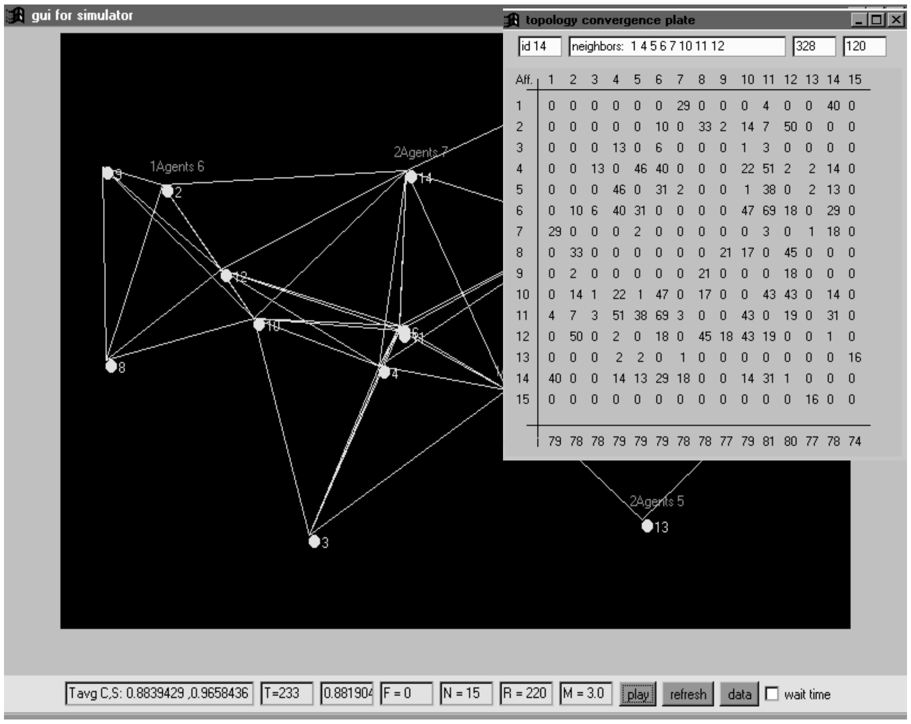
In step 1, the recency tokens of all the nodes stored in the information cache of the current host node and the recency tokens of all the nodes stored in the information cache of the agent is compared. If the recency token of any node, say X, in the host node's information cache happens to be less than that in the agent's information cache, then it is obvious that the agent is carrying more recent information about node X. So, the entire information about node X in the host node's information cache is overwritten by that in the information cache of the agent. This step is performed asynchronously for all the agents as they arrive at that host node. This step helps the node to acquire all the recent information that it can gather from the agents.

When an agent is ready to migrate (i.e. after a waiting time defined earlier as TtM), the step 2 is performed. In step 2, the agent copies the entire information cache of the host node on their own information cache. This contains the most recent information since the data set contains a combination of all the recent information that could be collected from the visiting agents and those that were already present in the node. With these updated values, the agents select their destination on the basis of navigation algorithm. The snap-shot in fugure 2 illustrates the result of the processes described in this section.

6. Information Aging: a Predictive Method for Topology Discovery

The foremost characteristic of a dynamic environment is that information is never absolute. Data sets gathered by nodes from agents on connectivity and link affinities are constantly changing. This might prove to be detrimental if not handled with care. For example, assume that a node receives information regarding the affinity of link x-y at time t_0. Now if it wants to start a data transfer session after time t_1, it just might happen that the link ceases to exist within that time. However, the node remains oblivious of this until another agent brings to it this information about the link disconnection. To avoid such events in a highly dynamic scenario, agent TtM might have to be decreased which will eventually increase agent-traffic in the network.

We have defined a concept of *information aging* on link-affinity based on which a predictive algorithm running on each node can predict the current network topology based on the current network information stored at that node. We apply this predictive mechanism to predict worst case topology and thus be cautious before data transfer is initiated. In our algorithm, at each time-tick, each node decreases the non-zero affinity values of all the links gathered at its information cache by a value equal to the average velocity per time tick. As indicated earlier, affinity is a worst-case prediction about the span of life 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 will accept the perception of node a about the link-affinity between n and m. We have already defined a metric called Average Link-affinity Convergence in order to evaluate the effectiveness of this scheme.



id 14 | neighbors: 1 4 5 6 7 10 11 12 | 328 | 120

Aff.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	29	0	0	0	4	0	0	40	0
2	0	0	0	0	0	10	0	33	2	14	7	50	0	0	0
3	0	0	0	13	0	6	0	0	0	1	3	0	0	0	0
4	0	0	13	0	46	40	0	0	0	22	51	2	2	14	0
5	0	0	0	46	0	31	2	0	0	1	38	0	2	13	0
6	0	10	6	40	31	0	0	0	0	47	69	18	0	29	0
7	29	0	0	0	2	0	0	0	0	0	3	0	1	18	0
8	0	33	0	0	0	0	0	0	21	17	0	45	0	0	0
9	0	2	0	0	0	0	0	21	0	0	0	18	0	0	0
10	0	14	1	22	1	47	0	17	0	0	43	43	0	14	0
11	4	7	3	51	38	69	3	0	0	43	0	19	0	31	0
12	0	50	0	2	0	18	0	45	18	43	19	0	0	1	0
13	0	0	0	2	2	0	1	0	0	0	0	0	0	0	16
14	40	0	0	14	13	29	18	0	0	14	31	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
	79	78	78	79	79	79	78	78	77	79	81	80	77	78	74

Tavg C,S: 0.8839429 ,0.9658436 | T=233 | 0.881904 | F = 0 | N = 15 | R = 220 | M = 3.0

Fig 2. 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 node is clicked. The window shows the information that a node with id=14 possesses about the network in the form of a 15 x 15 matrix, which contains the affinity values of links in 100 msec. unit. 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 have a low 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 = 1.52, mean = 78.26.

7. Performance Evaluation

7.1 Simulation Set-up

The proposed system is evaluated on a simulated environment under a variety of conditions to estimate the connectivity convergence and link-affinity convergence against time. In the simulation, the environment is assumed to be a closed area of 1500 x 1000 square meters in which mobile nodes are distributed randomly. We present simulation results for networks with 30 mobile hosts, operating at a transmission range of 400 m. The bandwidth for transmitting data is assumed to be 1 agents / msec. In order to study the delay, throughput and other time-related parameters, every simulated action is associated with a simulated clock. The clock period (time-tick) is assumed to be one millisecond (simulated). Agent TtM is assumed to be 50 or 100 msec.

The speed of movement of individual node ranges from 5 m/sec to 30 m/sec. Each node starts from a home location, selects a random location as its destination and moves with a uniform, predetermined velocity towards the destination. Once it reaches the destination, it waits there for a pre-specified amount of time, selects randomly another location and moves towards that. However, in the present study, we have assumed zero waiting time to analyze worst-case scenario.

7.2 Evaluating the Basic Mechanism

Fig. 3(a) 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. If the mobility is high, more links information would change per unit time. Thus it becomes more challenging for the agents to carry a link change information to all the nodes in the topology before the next link change takes place. As expected, performance varies directly with mobility. Lower the mobility better the results.

To compensate for increase in mobility, the solution would be to decrease agent TtM. It is evident from the graphs in fig. 3(b) that, for mobility 30 m/s, the performance is significantly different for the 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 increase. A predictive mechanism on network topology (as discussed in sec. 6) can achieve better result without lowering TtM.

7.3 Predictive Mechanism: Link-Affinity Convergence Analysis

We have analyzed the performance of our prediction mechanism and the results are presented in fig. 4(a) and (b). This mechanism ensures that the node is never misled to believe that a path exists for successful message transfer when the path might actually snap before that actual data is completely transmitted.

The Link-Affinity convergence graphs show satisfactory results as the curve always remains above 98 percent once the topology information has stabilized. This means that a link-affinity information that a node knows about a path has the probability that it is more than 98 % acceptable, even if the agent TtM is 100 msec.

7.4 Discussion

From the results in fig.3(a) it is clear that the average connectivity convergence improves with decrease in mobility. The performance goes below 80 % for a high mobility of 30 m/s. However, the time to migrate (TtM) could be lowered to produce better results even at high mobility (fig.3(b)). But this has the obvious effect of congestion in the system as the system sees more agent traffic in the medium per unit time. Our predictive mechanism (fig. 4) in the context of TtM=100 msec. could yield satisfactory results with the convergence values over 98 %. Thus we see that resorting to the predictive mechanisms even at a low agent migration frequency can uplift performance.

We look here into 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 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 $A / (\pi .R^2)$. 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,
$P_a = P.(\pi .R^2) / A$.

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 $[(T – t.P_a) *100 / T]$ % of the time. For example, if the bounded region of operation is 1500×1000 sq. m. and R is 400 m, and P and T are 15 and 100 milliseconds respectively for a 30 node network and t = 1msec., then $P_a = 5$ (approx.) and the medium would be free from agent traffic during 95 % of the time. In other words, for only 5 % of the total unit time, the medium gets blocked by agent traffic and the rest is free for data communication processes. This is the real gain in the case of the agent paradigm and serves as the major motivation to replace conventional flooding techniques.

8. Conclusion

In this study, we have designed a mobile, multi-agent based protocol to make the nodes in the network **topology aware.** We have assumed that agents do not get lost in transit or suffer from any kind of errors in transmission and reception. We have not considered the situations where a set of nodes switch themselves off after creating agents. In this situation, agent population in the network compared to number of active nodes would increase which would degrade the performance. In such a situation, some agents need to be destroyed. We have also not derived analytically the optimal value of TtM under a given condition which would ensure acceptable convergence keeping the agent traffic in the network tolerable. However, our preliminary results indicates that the use of mobile multi-agent framework would be able to make each node in the network topology aware without consuming large portion of network capacity.

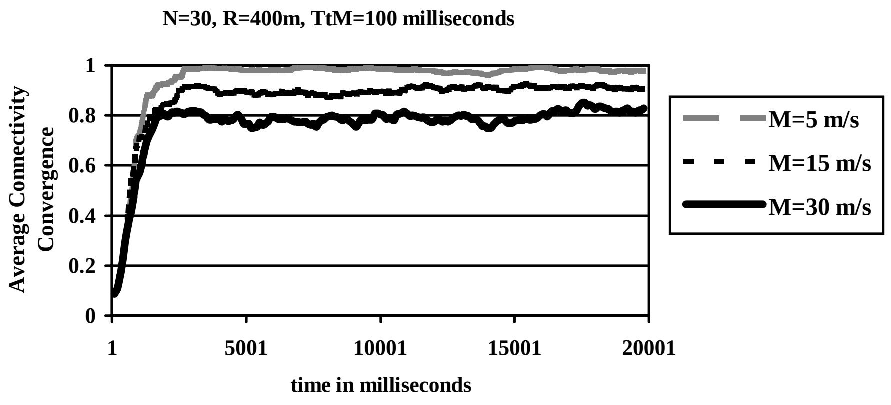


Fig. 3(a). Variation of connectivity convergence with time for different mobility ***<u>with TtM = 100 msec</u>.***

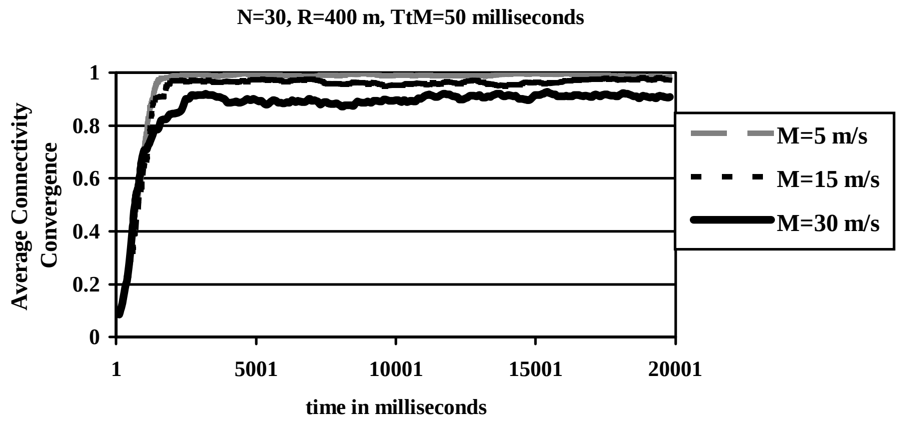


Fig. 3(b). Variation of connectivity convergence with time for different mobility ***<u>with TtM = 50 msec.</u>***

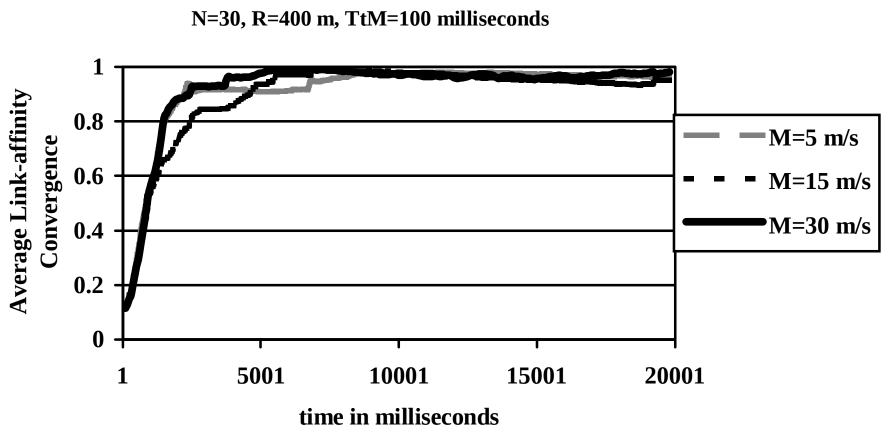


Fig. 4(a). Variation of Average Link-affinity Convergence with time for different mobility with TtM = 100 msec.

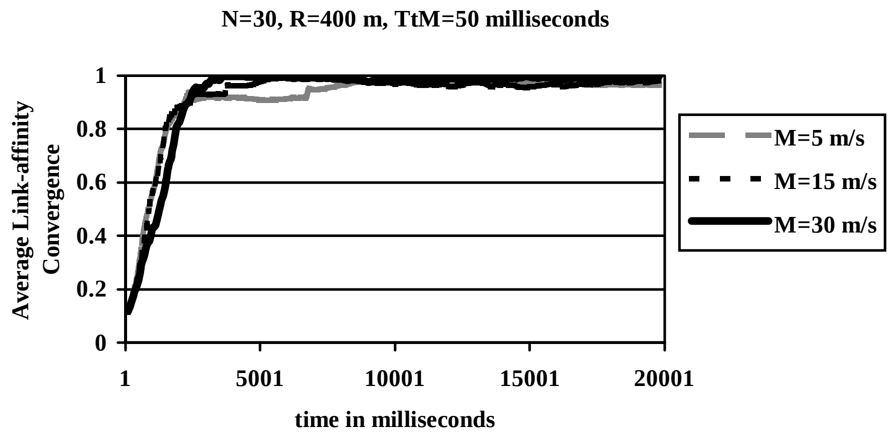


Fig. 4(b). Variation of Average Link-affinity Convergence with time for different mobility with TtM = 50 msec.

References

[1] Vu Anh Pham and Ahmed 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, 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] C-K Toh, A novel distributed routing protocol to support ad-hoc mobile computing, IEEE International Phoenix Conference on Computer & Communications (IPCCC'96).

[7] 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.

[8] Steve Appeleby and S.Steward. Mobile software agents for control in Telecommunications networks.*BT Technology Journal*,12(2):104-113,April 1994.

[9] T. Magedanz, K. Rothermel, and S. Krause, "Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?" Proc. INFOCOM'96, San Francisco, CA, 1996.

[10] M. Baldi, S. Gai, and G. P. Picco, "Exploiting Code Mobility in Decentralized and Flexible Network Management", K. Rothermel and R. Popescu-Zeletin, Eds., Mobile Agents, Lecture Notes in Comp. Sci. Series, vol. 1219, pp. 13–26, Springer, 1997.

[11] S. Krause and T. Magedanz, "Mobile Service Agents enabling Intelligence on Demand in Telecommunications", Proc. IEEE GLOBCOM '96, 1996.

[12] Schoonderwoerd, R., et al.(1997) 'Ant-based load balancing in telecommunications networks.' Adaptive Behavior,5(2):169207,1997. http://www.uk.hpl.hp.com/people/ruud/abc.html

[13] G.Di Caro and M.Dorigo.Mobile agents for adaptive routing.In Proceedings of the 31st*Hawaii International Conference on Systems*, January 1988.

[14] Nelson Minar, Kwindle Hultman Kramer and Pattie Maes.Cooperating Mobile Agents for Dynamic Network Routing.In Alex Hayzeldon, editor, *Software Agents for Future Comunications Systems*, chapter 12.Springer-Verlag, 1999. http://www.media.mit.edu/~nelson/research/routes-bookchapter/

[15] S. Bandyopadhyay and Krishna 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

[16] Jonathan 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.