Hybrid Simulation for Routing Using Ant Colony with Queuing Analysis Algorithms

Bhanu Pratap Singh¹, Dr. Sohan Garg²
¹Research Scholar Mewar University-Ghaziabad, Uttar Pradesh, INDIA
²Director-SCRIET Meerut, Uttar Pradesh, INDIA

I. INTRODUCTION

Ad hoc networks can operate without fixed infrastructure and survive rapid changes in the network topology. Usually nodes are mobile and use wireless communication links. In recent years, many routing algorithms for ad hoc networks have been proposed. The algorithms are most often compared using simulation. There are three different ways to model and evaluate networks; formal analysis, real life measurements and simulation [Särelä, 2004]. The dynamic nature of ad hoc networks makes them hard to be studied by formal analysis. Some formal techniques that have been used in static networks include Petri nets, stochastic processes, queuing theory and graph theory. Since ad hoc networks are still mainly a research subject, most scenarios that will be used in are still unknown. One of those scenarios that are known is military networks. Thus, use of real life measurements is currently almost impossible and certainly costly. The commonly used alternative is to study the behavior of the protocols in a simulated environment. The purpose of simulation is to create an artificial environment, usually a computer program, that captures the essential characteristics of the phenomena that is being studied. Simulation is an economically viable way to create a statistically significant amount of test runs. For these reasons, simulation is an economically viable way to create a statistically significant amount of test runs. The suggested model to minimize end-to-end delay in a MANET [Tarek and Mueller-Clostermann, 2004].

II. A COMPLETE SCENARIO OF THE SIMULATION ALGORITHM

The main goal of this paper is to present a novel complete hybrid simulation model for routing in Mobile Ad hoc Networks (MANETs) inspired by Ant Algorithm in combination with queuing network analysis. The routing algorithm is based on a type of learning algorithm, similar to one described in AntNet, that provides deterministic forwarding of message packets (ant packets) from source node to destination node. Our algorithm assumes that all links in the network are bi-directional and all the nodes in the network fully cooperate in the operation of the algorithm.
Figure 1. Routing table structure imbedded in a network node

Figure 2. An example of topology with varying connectivity

Let’s consider an example of a routing table. Table 1 is the routing table of node 5 from the network shown in Figure 2. Because the neighbor nodes of node 5 are the nodes 2 and 4, there exists a column for each of them. For every other node, except node 5, there is a row in the routing table. The probability to reach node 2 in shortest time by directly going to node 2 (Pnode2, node2) is very high, namely 0.95. It is also possible to go to node 2 along node 4, but the probability to go there in shortest time (P node2, node4) is very small, only 0.05. The probabilities to go to node 1 in shortest time do not deviate so strongly from each other. (P node1, node2) is 0.59 and (P node1, node4) is 0.41. That is because the route to node 1 along node 2 is not much shorter than the route to node 1 along node.

Table 1. Routing table of node 5 from the network of Figure 5-8

<table>
<thead>
<tr>
<th>Destination node</th>
<th>Next node</th>
<th>node 2</th>
<th>node 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>node 1</td>
<td></td>
<td>0.59</td>
<td>0.41</td>
</tr>
<tr>
<td>node 2</td>
<td></td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>node 3</td>
<td></td>
<td>0.27</td>
<td>0.73</td>
</tr>
<tr>
<td>node 4</td>
<td></td>
<td>0.06</td>
<td>0.94</td>
</tr>
</tbody>
</table>

The probabilities in the routing tables can be compared with the strength of the pheromone trail. The higher the probability the stronger the pheromone trail. Note that the routing tables only contain local information on the best routes and no global information. To compute the shortest routes in time needed to send a packet from one node in the network to another node, the information of all routing tables on the path from source to destination node is needed. The next step from a node toward the destination node is always determined by using the routing table of the node. The next node is the node with highest probability in the row that represents the destination. By that, packets are always routed to their destination along the path with highest probability. Since this path represents the strongest pheromone trail to the destination, it should be the shortest known path in time.

2.2 Queuing Network Analysis

Kleinrock [Kleinrock, 1964] [Wong, 1978] first derived an expression for the mean End-to-End delay in a message-switched network. He chose to model a data network as a network of communication channels whose purpose was to move data messages from their origin to their destination. Each channel was modeled as a server serving a queue of data messages awaiting transmission. The main metric used for the performance of the network was $T$, the average time messages took to move across the network. One of the first general results was an exact expression for the mean delay experienced by a message as it passed through a network. The evaluation of this delay required the introduction of an assumption (Kleinrock’s Independence Assumption) without which the analysis remains intractable, and with which the analysis becomes quite straightforward. The independence assumption assumes that “the length of a message is chosen independently from the exponential distribution each time it enters a switching node in the computer network”. In our hybrid simulation, we will use Kleinrock’s Independence Assumption to calculate the delay of sending a packet through a mobile ad hoc network. The End-to-End delay is defined as time between the moment in which the source wants to send a packet and the moment the packet reaches its destination.

\[
\text{End-to-End delay} = T_{\text{destination receives packet}} - T_{\text{source wants to send packet}}
\]

The End-to-End delay is important because nowadays, many applications (e.g. IP telephony) need a small latency to deliver usable results. It shows the suitability of the protocol for these applications. The expected response time delay to send packets from a source node $S$ to a destination node $D$ is the sum over the response times at all links and nodes visited along the way [Haverkort 1998]:
2.3 Ant Algorithm

This section describes ant algorithm in detail. Firstly, we will explain the intelligent agents that will be used in the ant algorithm.

What has not been considered so far is how the probabilities in the routing tables are computed and updated. In the nature, ants lay pheromone and so they produce pheromone trails between the nest and a food source. On a computer, the pheromone has been replaced by artificial stigmergy, the probabilities in the routing tables. To compute and update the probabilities, intelligent agents are introduced to replace the ants. There exist two kinds of agents, the forward agents and the backward agents. All forward and backward agents have the same structure. The agents move inside the network by hopping at every time step from a node to the next node along the existing links. The agents communicate with each other in an indirect way by concurrently reading and writing the routing tables on their way. The forward and backward agents receive percepts from the environment so they can do certain actions. Condition action rules are used to define what action should be chosen under which situation. The current situation is defined by the percept and the stored internal state. The internal state of the agents is a kind of memory which stores a list of \((k, tk)\) pairs. Every such pair represents a node that has been visited by the forward agent. \(k\) is the identifier of the visited node and \(tk\) is the time a packet takes to travel the link from the last visited node to this node under the current traffic situation.

2.3.1 Forward Ants

Each node \(s\) periodically sends a forward ant (forward agent) \(F_s \rightarrow d\) to a randomly chosen destination node \(d\) throughout the network. The task of the forward ant is to discover a feasible, low-cost path to the destination and to gather useful information on its trip.

At every visited node \(k\) on the way to the destination node, a forward ant does the following:

- The forward ant checks its stack memory whether node \(k\) has already been visited before or not. If it has been visited, there will be a cycle in the ant’s path and this cycle will be deleted from the memory. The cycle’s nodes are popped from the ant’s stack and all the memory about them is destroyed as indicated in Figure (5b).
- The forward ant updates its memory by adding a new \((k, tk)\) pair to the memory as indicated in Figure (5a).
- If node \(k\) is not the destination node then the forward ant determines the next node to go to by using the probabilities in the row of the routing table of node \(k\) which represents the destination node.
the following technique: The individuals (nodes of each routing table) are mapped to contiguous segments of a line, such that each node’s segment is equal in size to its fitness (probability of routing). A random number between 0 and 1 is generated for every link to a neighbor node according to the magnitude of the probabilities and the node whose segment spans the random number is selected. This technique is analogous to a roulette wheel with each slice proportional in size to the fitness.

Table 2 shows an example of the next hop selection process from the routing table at a mobile node E. We remark that, the mobile node C has the highest probability to be chosen as a next hop node and occupies the largest interval, whereas node G is the least probability node that has the smallest interval on the line as shown in Figure 5-12.

<table>
<thead>
<tr>
<th>Next hop</th>
<th>D</th>
<th>G</th>
<th>F</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probability of routing</td>
<td>0.16</td>
<td>0.14</td>
<td>0.25</td>
<td>0.45</td>
</tr>
<tr>
<td>Cumulative probability</td>
<td>0.16</td>
<td>0.30</td>
<td>0.55</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 2. Selection probability at mobile node E

Figure 5-14 shows the selection process of the new nodes. For example, if the generated random number is 0.73, then node C will be selected as a next hop.

Figure 6. Roulette wheel selection

The individuals are selected according to their probabilities. The higher the probability, the higher the probability of being selected. The roulette can be described with the following algorithm:

1. **[Sum]** Calculate sum $S$ of all neighbor nodes probabilities for a node $k$.
2. **[Select]** Generate random number $r$ from the interval $(0, S)$.
3. **[Loop]** Go through the neighbor nodes and sum the probabilities from 0. When the value of this sum exceeds the value of $r$, the procedure stops and the current node becomes the selected one.

Now, we go back to our discussion about the ant algorithm. The node where the ant has just come from is filtered out to avoid that the ant directly goes back to that node. With the generated probabilities, the next link is randomly selected. The forward ant goes to the next node along that link. If node $k$ is the destination node then the forward ant transforms to a backward ant $B_{y+s}$.

Now, all forward ants arrived at their destination node. Each ant has a stack $Ss\rightarrow d(k)$ of the virtual elapsed time for each ant trip. The stack of the forward ant is a dynamically growing data structure that contains the id number of the nodes that the forward ant has traversed as well as the elapsed time between its starting from $s$ to its arriving to node $k$. For example, the stack for one of the ant trips from source $s$ to destination node $d$ shown in Figure 5-11a. The trip time to reach the desired neighbor is computed using the formula:

$$dij + (qij + Sa)/ Bij$$

Where $dij$ is the link’s propagation delay (distance/signal propagation speed) between two mobile nodes $i$ and $j$. Note that this value can be neglected because the distance value is very small in comparison to the value of signal propagation speed. Forward ants are routed on normal priority queues, that means, they use the same queues as normal data packets. As such, forward ants face the same network conditions (queuing and processing delays, network congestion) as data packets. Forward ants therefore contain information regarding the route that they have traversed. We can summarize the work of the forward ants as follows; each forward ant selects the next hop node using the information stored in the routing table. The route is selected, following a random scheme, proportionally to the goodness (probability) of each path and to the local queues status. In case the selected link is not currently available, the forward ant waits its turn in the low-priority queue of the data packets, where it is served on the basis of a FIFO policy.

### 2.3.2 Backward Ants

The backward ant (backward agent) inherits the memory from the forward ant. The task of the backward ant is to go back to the source node $s$ along the same path as the forward agent but in the opposite direction and to update the routing tables on this path. At every visited node $k$ on the way back to the source node a backward agent does the following:

- The backward agent updates the routing table of node $k$ by using the travel times stored in its memory. The update process is explained in the next section.
- If node $k$ is not the source node then the backward agent uses its memory to determine the next link of the path back to the source node. The backward agent goes to the next node along that link.
- If node $k$ is the source node then the backward agent is killed.

Let’s summarize the indirect communication of...
the intelligent agents via artificial stigmergy. The forward agents use the routing tables (artificial stigmergy) of the nodes to find their way to the destination node. They only have reading access to the routing tables and can not change the probabilities. The backward agents use their memory to find the way back to the source node. On their way back, they update the routing tables by changing some of the probabilities. By that, the probabilities in the routing tables change very fast, possibly several times a second. The next forward agents that visit the nodes will find other values in the routing tables and therefore other conditions under which they react. By this indirect communication between the forward ants and the backward ants, an emergent behavior of the quite simple ants arises like in ant colonies that leads to optimize routes.

2.3.3 Updating Routing Tables

Updating in routing tables at each node will be done by backward ants using ants trip times. At every visited node \( k \) on the way back to the source node, a backward ant updates some of the probabilities in the routing table of node \( k \) by using the travel information in its memory which was collected by the forward ant.

2.4 Performance Evaluation

In this section, we evaluate the End-to-End delay \( E[V] \) for the packets through the MANET using the framework described in the preceding sections.

2.4.1 Simulation Model

During the simulation, nodes are free to move anywhere within this area. Each node travels toward a random spot, then takes a rest period of time in second. After the rest period, the node travels toward another randomly selected spot. This process repeats throughout the simulation, causing continuous changes in the topology of the underlying network, followed by a simulation of the ant behavior yielding an improvement of the routing tables, which is evaluated by Kleinrock’s delay analysis. Finally, we get the minimum delay from source to destination node. The simulation program has been executed on standard 350 Mhz PC using Visual Basic 6.0. It needs few seconds of CPU time for a single simulation run. The model parameters that have been used in the following experiments are summarized in Table 5-4.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of nodes</td>
<td>10, 20, ..., 150</td>
</tr>
<tr>
<td>Arrival rate</td>
<td>150 Kbps</td>
</tr>
<tr>
<td>Transmission range</td>
<td>250 m</td>
</tr>
<tr>
<td>Velocity / Direction</td>
<td>10 m/sec &amp; 45 degree</td>
</tr>
<tr>
<td>Packet size</td>
<td>64 byte</td>
</tr>
<tr>
<td>Pause time (Ant time)</td>
<td>10, 20, ..., 100 sec.</td>
</tr>
<tr>
<td>Simulation time</td>
<td>180 sec</td>
</tr>
<tr>
<td>Link bandwidth</td>
<td>1 Mbps</td>
</tr>
<tr>
<td>Simulation area</td>
<td>1000 m × 800 m</td>
</tr>
<tr>
<td>Mobility model</td>
<td>RWM &amp; BSAM models</td>
</tr>
<tr>
<td>Routing protocol</td>
<td>Ant algorithm</td>
</tr>
</tbody>
</table>

Table 5-4. Simulation parameters

We will use a simple scenario for evaluating our ad hoc routing algorithm with the following assumptions:

- 2-d rectangular area
- No obstacles
- Bi-directional links
- Fixed number of nodes
- Nodes operate for whole simulation
- Random waypoint mobility (RWM) model: Nodes pause for a random amount of time before picking uniformly distributed points from the simulation area. Nodes then move to new points in the simulation area.
- Boundless simulation area mobility (BSAM) model: There exists a relationship between the previous and the current movement direction \( \theta \) and velocity \( v \). Both direction and velocity are deterministic through the whole simulation time.

2.4.2 Experimental Results

The first experiment shows the relation between increasing the number of nodes and the End-to-End delay in case of using different numbers of nodes (10, 20, ..., 140, 150) while ant time is constant at 30 sec.

![Figure 7. Number of nodes vs. mean delay](image)

Figure 7 shows that increasing the number of nodes results in an increase in the delay, because each hop can contribute a substantial amount of delay in forwarding traffic. Furthermore, the more nodes, the more congestion and the longer it takes to discover routes. The second experiment investigates the relation between increasing the ant time and the End-to-End delay in case of using different pause times (10, 20, ..., 100 seconds) and 60 nodes.

![Figure 8. Ant time vs. mean delay](image)
Figure 8 shows that increasing the pause time leads to a decrease in the delay, because the ant algorithm performs more iterations that help to approach the minimum delay. The third experiment is done for a scenario with 60 nodes distributed randomly and an ant time of 60 seconds. The whole simulation time is 180 seconds and we observe the packet delays over this time period. Figure 9 shows the mean delay for the first topology. Figures 5-16 and 5-17 show the mean delay for the topologies, which has been established from the preceding topology by applying the mobility model. These figures do display 95% confidence intervals.

![Figure 9. Mean delay through first topology](image)

Figure 10 summarizes the tracing of the delay over the whole simulation time of 180 seconds for three different topologies. We note from the figure that there is no significant effect on the delay between RWM and BSAM models, but the performance can vary significantly with other different mobility models as stated in [Camp, 2002].

![Figure 10. Mean delay through second topology](image)

After a change in the topology, we observe a degradation of the performance (i.e. an increase of the delay) followed by a performance improvement due to the work of the ant algorithm which converges step-by-step toward the minimum possible delay. The last experiment is done to remark the relation between the number of nodes and loop detection time. Figure 5-19 indicates that the increasing in the number of nodes will cause increasing in the mean time for loop detection [Tarek, 2004].

![Figure 12. Packet delay over 180 sec. simulation time](image)

The higher the node density, the higher the possibility of loops formation, which in turn causes the time of loop detection and deletion to increase.

**REFERENCES**


