CHAPTER 6

MTA IMPLEMENTATION ON MANET WITH QOS SPECIFIED EFFICIENT LOCAL ROUTE REPAIR STRATEGY

Due to dependency of telecommunications networks on information processing, network resiliency has become a critical research area. Network resiliency helps in judging the ability of the network to support Quality of Service (QoS) to customers even in a highly dynamic topology where the link/node failure is very frequent. Network resiliency includes methodologies like prediction based route repair and efficient utilization of spare resources. These two methodologies guarantee seamless communication upon failure. Most of the failures are difficult to predict and eliminate, but it is possible to reduce the impact of set of specific failure by introducing network resilience techniques into the network design phase [98,100].

Wireless networks such as Mobile AdHoc Networks (MANETs) have many advantages compared to wired networks. In MANETs the communication is not limited to a certain geometrical region. Swarm Intelligence based ACO algorithms provide interesting solutions to network routing problems. ACO based routing in MANETs will enhance the reliability and efficient packet delivery. They help in reducing control overhead due to their inherent scalable feature [99].

This research focuses on the development of a robust Swarm Intelligence system for MANETs. However due to certain challenging characteristic of MANETs; it is vulnerable to lose packets or gets congested. The main causes of these characteristics are the dynamic nature of the topology. Due to mobility of nodes in MANETs routes between source and destination have to be adjusted dynamically. Therefore route maintenance and route discovery modules have to be very efficient. These two modules help MANETs to discover paths to the destination and find valid routes when there is link/node failure.
The Termite algorithm contains several tunable parameters and methods to automate the selection of optimal routes for different network conditions. However, Termite algorithm does not contain methods for determination of QoS, Route Maintenance; Load balancing etc. Modified Termite Algorithm (MTA) is proposed with a prediction mechanism that anticipates link breaks and repairs them before they break, thereby avoiding unnecessary warning messages and reducing control overhead. Segmentation of data packets is provided, that need to be transmitted from the source to the destination i.e. transmission of data is done through an alternative path to the destination which had not been transmitted earlier prior to link failure through the failed route. Algorithm also provides QoS for providing the best effort delivery and transmits its data only through those paths which can satisfy the required bandwidth.

By including QoS, efficient route maintenance and local repair strategy by prediction of node failures, the MTA is expected to enhance the performance of the network in terms of throughput and reduction of end-to-end delay and routing overheads [105,106,108].

6.1 Modified Termite Algorithm with QoS

The Termite algorithm uses the traditional flooding optimizations and will result in a lower control packet overhead. The application is designed for the best effort delivery of data. The aim of the application is to build an optimal path from source to destination and maintain it by satisfying the available bandwidth. It randomly generates multiple paths and also performs load balancing by selecting an alternate path, in case the optimal path gets congested.

In traditional Termite algorithm, route repair takes lot of time to find an alternate path for the destination or the source. After waiting for some time it will generate a fresh route discovery. This procedure takes a lot of control overhead. Therefore a new methodology in which preemption based Local route repair strategy is adopted to overcome the problem of control overhead.
6.1.1 Route Discovery

The application has three phases which are Route Discovery, Route Maintenance and Route Failure Handling. The Route Discovery is responsible for providing Quality of Service (QoS) between a given source and a given set of reachable destinations. Route Request (RREQ) performs a walk over the network until it reaches the requested destination. If the bandwidth is available then the route request packet is forwarded towards the destination node as shown in Figure 6.1. While traveling to the destination the Route Request collects the available capacity of each link, the number of hops visited. Then it will respond with a Request Reply (RREP) packet, which will be initialized with the available bandwidth value [114,115].

**Sequence Diagram**

![Sequence Diagram of Route Discovery with QoS](image)

**Fig 6.1** Sequence diagram of route discovery with QoS

6.2 Efficient route maintenance by predictive preemptive local route repair strategy

In most of the existing protocols, a node that detects the failure of a link to a destination propagates a link failure control packet to all upstream (towards sources) neighbors. This process is repeated until all sources that use the link are notified. Source nodes then invoke the route discovery mechanism if they need a route to the destination.
6.2.1 Failure Handling

This module takes care of link failures, power issues and it runs all the time. A node is in an unsafe or preemptive region if the power of the signal it sends to its successor node is below a threshold power. Consecutive measurements of the signal strength of packets it sends to the successor node are done and link failure is predicted using the Lagrange interpolation formula. If the interpolated signal strength so obtained is less than the minimum acceptable power then the node is said to have failed. The node has to inform its predecessor neighbor of its failure as shown in Figure 6.2.

This module also takes care of battery power of the node which is considered to decrease linearly as well as due to the processing of the packets. When the battery power goes below the minimum acceptable power then the node is said to have failed and has to inform its predecessor neighbor about its failure.

**Fig 6.2 Failure Handling Module in MTA**

In this approach, the Lagrange interpolation for estimating whether an active link to a neighboring node will fail is used. When link failure is expected between a node and an upstream neighbor, the upstream neighbor itself first attempts to find a route to the
destination. If such a route is not found within a discovery period, a link failure warning is propagated via upstream nodes to sources that use this link. Source nodes then invoke the route discovery mechanism if they need a route to the destination. The main goal of this approach is to avoid sending unnecessary warning messages thus reducing control overhead. A node is considered to be in an unsafe or preemptive region, if the battery power of the node drops below a pre-defined threshold power. The power of each node is continuously monitored and link failure is predicted using Lagrange interpolation. Lagrange Interpolation generates a random value for each node (in the form of battery power) and the power of the node decreases by that factor continuously in time. When the power of a node becomes lower than the minimum acceptable power, a warning message is sent to the predecessor node which then attempts to find an alternative path to the destination.

6.2.2 Failure Recovery

This module takes care of failure recovery by local repair. When an upstream neighbor gets a warning message from a failing successor node, it stores the sequence number of the first packet and discards the rest of the queued packets stored in the buffer. It sends a sleep signal to the source so that it freezes its transmission of packets. It starts rediscovery to find an alternate route to the destination. If the upstream neighbor is successful in finding an alternate path to the destination then it sends a signal to the source informing it to transmit the packets from the sequence number it had stored. It continues the transmission of the rest of the packets in the newly found path.

If the upstream neighbor fails in finding an alternate path to the destination, it informs its predecessor node, the sequence number and the hop value. Now it is the responsibility of this predecessor node to discover the route. This process goes on till a minimum no. of hops is reached, after which the source is informed of the failure of the local repair process.

Route maintenance phase basically helps in strengthening the route which has already been established during the route discovery phase as shown in Figure 6.3. The strengthening of the route is done by the data packets. Every time a data packet
travels a particular link, it increases the pheromone value of that link in the respective nodes [116].

Route maintenance phase not only strengthens the path but also performs the periodic decay of the established path in order to search for a new optimal path which might have changed due to mobility of nodes. This part of the phase gives a dynamic nature to the algorithm [117,118].

**Proposed Algorithm**

**Step1:** Use Lagrange interpolation method (which generates random numbers in each node). This is used as analogy for the battery power left in each node to continuously determine the power of each node in the established network.

\[
P = \sum_{j=0}^{n} \left[ \prod_{j=0}^{n} (t - t_j) \times p_j \right] \times \frac{n}{\prod_{j=0}^{n} (t_i - t_j)} \quad \text{.......................... (7.1)}
\]

Fig 6.3 Failure Recovery Module in MTA
Lagrange Interpolation generates a random value. This value in simulation helps us to generate battery power left in each node. Every node’s battery power decreases continuously in time as and when a packet is processed by the node. The expected power $P$ of the node is computed as follows:

$$P = \left[ \frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} \times P_0 \right] + \left[ \frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} \times P_1 \right] + \left[ \frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)} \times P_2 \right]$$

Where $P_0$, $P_1$, $P_2$ - Are measured power strengths at the measurement times $t_0$, $t_1$ and $t_2$, respectively.

**Step2:** When the power $P$ of a node $N$ is lower than the minimum acceptable power, hello module is killed and a warning message is first propagated to the predecessor node and in case the predecessor node is unable to find an alternative path to the destination, then the link failure warning message is further propagated to all upstream sources that make use of this node. The routing table is also updated to notify the nodes in the network about the change in the network topology that is expected to take place as a result of node failure.

Predict ()
{
    If (P <= Min_Acc_Power) then
    {
        Send_Warning (predecessor_node);
        Local_route_repair ();
        //transmit only the remaining data packets which had not been transmitted earlier //through this route, prior to link failure, through an alternative route, to the //destination
    }
}
Step3: The predecessor of node N then initiates a local route repair procedure to find an alternative path to the destination by consulting the updated routing table. The source sends only the remaining data packets which had not been transmitted earlier through the path in which the link failure had occurred thereby leading to its abandoning, through an alternative route to the destination.

Step4: When the power of the node N becomes zero, then the node is removed from the network and all links attached to it are broken.

6.3 Results and Discussion

Some of the improvements to Termite algorithms could be reserving resources in the form of time slots, bandwidth and efficient local route repair strategy. The present work focuses on development of an efficient routing algorithm ‘Modified Termite Algorithm’ (MTA) using termites based on ACO and implementation on MANET. The primary metrics considered for the algorithm are Throughput, End-to-End delay and routing over head against the exhausted nodes.

A series of tests are conducted to check the feasibility of MTA as an effective mechanism for MANETs based on the community based mobility model. The behavior of success rates with the number of termites (nodes) has been studied. It is observed that only after 30 nodes, the algorithm shows a sharp increase in the success rate (considering 5 m/s mobility speed and pause time varying between 1-5 secs).

Using MTA with QoS and with effective local route repair strategy detailed analysis has been carried out on MANETs to determine the throughput, end-to-end delay and routing overhead by varying the number of exhausted nodes (nodes whose battery power has reached the threshold). The throughput derived as a function of increase in the number of exhausted nodes from 1-7 for MTA with 30 nodes is compared with Termite algorithm and shown in Figure 6.4.
Figure 6.4 shows that for MTA, there is a continuous increase in throughput with the decrease in number of exhausted Nodes from 1 to 7, whereas the throughput for Termite algorithm remains low. This suggests that the MTA algorithm with 30 nodes indicates significant improvement in the throughput ~16% with the increase in number of exhausted nodes as compared to Termite algorithms.

Similarly the throughput derived as a function of increase in the number of exhausted nodes for MTA with 50 nodes is compared with termite algorithm and shown in Figure 6.5.

In Figure 6.5 the MTA based MANET shows continuous increase in throughput with the decrease in number of exhausted nodes, whereas the throughput for Termite algorithm is. This suggests that the MTA algorithm implemented on a MANET with 50 nodes indicates significant improvement in the throughput ~3% with the increase in number of exhausted nodes as compared to termite algorithms.

It may be noted that the MTA algorithm exhibits ~12 % improvement in the throughput performance for lower number of 30 nodes compared to 50 nodes.
The MTA with 30 nodes implemented on MANET has been used to determine the end–to-end time delay with the increase in number of exhausted nodes from 1-7. Figure 6.6 shows the end–to-end delay variations for an increase in number of exhausted nodes. In the same figure the end to end delay variations for the termite algorithm which do not incorporate efficient pheromone decay technique are also shown for comparison.
In Figure 6.6, the Termite algorithm shows continuous increase in end-to-end time delay with the increase in number of exhausted Nodes, whereas the end-to-end time delay for MTA suggests that the MTA algorithm implemented on a MANET with 30 nodes indicates significant reduction in the end-to-end time delay ~30% compared to Termite algorithms. Similarly end-to-end time Delay with increase in the number of exhausted nodes from 1-7 for MTA with 50 nodes is compared with the Termite algorithm and shown in Figure 6.7.
In Figure 6.7, the Termite Algorithm shows continuous increase in end-to-end time Delay with the increase in number of exhausted Nodes, whereas the end-to-end time delay for MTA suggests that the MTA algorithm implemented on a MANET with 50 nodes indicates reduction in the end-to-end time delay ~26% times compared to Termite algorithms.

It may be noted that the MTA algorithm exhibits ~4% significant reduction in the end-to-end time delay for lower number of 30 nodes compared to 50 nodes.

The MTA with 30 nodes has been used to determine the routing overhead changes with the increase in number of exhausted nodes. Routing overhead is the ratio of number of control packets required for delivery of data. Figure 6.8 shows the routing overhead changes for increase in number of exhausted nodes. In the same figure the routing overhead changes for the termite algorithm is shown for comparison.

![ROH Vs Exhausted Nodes](image)

**Fig 6.8 Routing Overhead Vs No of Exhausted Nodes**

In Figure 6.8, the Termite algorithm shows continuous increase in routing over heads with the increase in number of exhausted nodes, whereas the routing over heads for MTA suggests that the MTA algorithm implemented on a MANET with 30 nodes
indicates significant decrease in the routing overheads ~2% compared to Termite algorithms.

Similarly routing overhead with increase in the number of exhausted nodes for MTA with 50 nodes is compared with termite algorithm and shown in Figure 6.9.

![ROH Vs Exhausted Nodes](image)

**Fig 6.9 Routing Overhead Vs No of Exhausted Nodes**

In Figure 6.9, the Termite algorithm shows continuous increase in routing over heads with the increase in number of exhausted nodes, whereas the routing over heads for MTA suggests that the MTA algorithm implemented on a MANET with 50 nodes indicates significant decrease in the routing overheads ~1.9 % compared to Termite algorithms.

It may be noted that the MTA algorithm exhibits any significant variation ~0.1 % in the Routing over heads for lower number of 30 nodes as compared to 50 nodes.

The modified Termite algorithm (MTA) with fine tuning of pheromone concentration shows a significant increase in the throughput, a large decrease in end to end delay and small decrease in routing overheads with the exhaustion of nodes. The reduction of routing overheads can be used for achieving the local route repair strategy. Further the local route repair strategy will improve the QoS by efficient bandwidth utilization.
The MTA developed by adopting efficient pheromone evaporation technique and by including QoS, efficient route maintenance, and local repair strategy enhances the performance of the MANET in terms of throughput, and reduction of end-to-end delay and routing overheads.