CHAPTER 2

DISTRIBUTED ALGORITHMS FOR MANET

This chapter presents two distributed algorithms (Local Recency & Local Frequency) for adapting the topological changes in ad-hoc networks by circulating the token through all the nodes of a mobile ad-hoc network. An important application is to ensure the group communication; the total order of message delivery is used in the distributed system.

2.1 INTRODUCTION

In a MANET, communication links may cause a danger impact during communication because of the mobility of the nodes. This nature made significant changes in the network links, the capacity of the network and consists of multiple nodes that can roam anywhere inside the network. While developing an algorithm for adjusting these kinds of problems in MANET distributed models. However, these problems have not been studied as extensively as their corresponding versions in more traditional parallel or distributed models. Rather, much of the work on MANETs has focused on lower-level issues, such as routing and media access control (MAC). It is one of the main protocols of data link layer. As such, the study of the higher-level problems in MANETs is still an active research area with several opportunities for contribution. Problems such as leader election, mutual exclusion, and consensus, have been extensively studied in many different traditional parallel and distributed computing models. Many of the problems that have been extensively studied in parallel or distributed computing models
have natural analogs in a relatively new model: the Mobile Ad-hoc Network (MANET).

Leader election is an extensively studied problem in distributed systems and is a useful building block in such systems, especially in environments (like Mobile Ad-hoc Networks) where failures may occur. Leader election can be used to assist with mutual exclusion, group communication, and other standard problems in distributed systems. Additionally, solving leader election in MANETs is useful for other reasons, such as assisting in complex motion planning algorithms and serving as a building block for creating particular tree communication structures. The traditional definition of leader election in static distributed systems requires that the processors in the system eventually elect a unique leader. In a MANET, though, link changes are common and may cause the network to split into multiple connected components or cause some components to become separated from the leader. Additionally, two connected components, each with its own leader, may merge. Thus, the definition of the leader election problem has to be adapted to the Mobile Ad-hoc environment. As in we define the leader election problem in MANETs as the problem of guaranteeing that any component of the mobile ad hoc network whose topology is static sufficiently long will eventually have exactly one leader. The main objective of this work is that using local recency and local frequency algorithms the tokens are circulated around the distributed nodes. It reduces the token circulating time. The designation node receives the message as early as possible. Furthermore, if the leader moves away or fails then the neighbor nodes have to enable conducting an election and choose their leader. It improves the mobile ad hoc network nodes stability.

This chapter organized as follows. Section 2.1 presented basics on distribution in MANET and section 2.2 provides some related works in this
area. Section 2.3 describes our proposed works and section 2.4 explains the implementation. Section 2.5 provides the results. Finally, we conclude our work in section 2.6.

2.2 RELATED WORKS

The accommodation of mobile hosts (MHs) within existing data networks, the static network is augmented with “mobile support stations” (MSSs) that communicate directly with MHs, usually via wireless links (Acharya & Badrinath 1996). Connectivity of the overall network changes dynamically as MHs connects to the static network from different “locations” (MSSs) at different times. Compared to their desktop counterparts, mobile hosts face a new set of constraints namely, low bandwidth of the wireless links, tight constraints on power consumption and a significantly lower computing capability. Thus, even without considering failures, integration of mobile computers within existing networks pose a new set of problems. In this chapter, we look at the problems associated with delivering multicast messages to mobile hosts. Next, we introduce “multicast groups” of mobile hosts wherein each multicast group is associated with a “host view” a set of MSSs representing the aggregate location information of the group. A host-view membership algorithm is then presented and combined with the multicast scheme for exactly-once delivery. As a result, to deliver a multicast message to a specified group, copies of the message need be propagated only to the MSSs in the group’s host-view.

The group communication services are becoming accepted as effective building blocks for the construction of fault-tolerant distributed applications (Amir et al. 1992). Many specifications for group communication services have been proposed. However, there is still no agreements about what these specifications should say, especially in cases where the services are partitionable, i.e., where communication failures may lead to the
simultaneous creation of groups with disjoint memberships, such that each group is unaware of the existence of any other group. In their work, they present a new, succinct specification for a view-oriented partitionable group communication service. The service associates each message with a particular view of the group membership. All send and receive events for a message occurs within the associated view. The service provides a total order on the messages within each view, and each processor receives a prefix of this order.

The token protocol supports consistent concurrent operations by placing a total order on broadcast messages (Amir et al. 1993). This total order is achieved by including a sequence number in a token circulated around a logical ring that is imposed on a set of processors in a broadcast domain. A membership algorithm handles reconfiguration, including restarting of a failed processor and re-merging of a partitioned network. Effective flow-control allows the protocol to achieve message-ordering rates two to three times higher than the best prior protocols. The single-ring total ordering protocol of Totem provides fault-tolerant agreed and safe delivery of messages within a broadcast domain.

2.3 GROUP COMMUNICATION SERVICE FEATURES

The advantages of a group communication service are

i. Maintains group membership for the neighboring nodes.

ii. Group communication with event ordering.

Group communication services have been successfully used in the past as building blocks and abstractions for implementing distributed tasks. Past work on total ordering has yielded several approaches, which use a token to implement the total order. The group communication problem becomes
difficult in a MANET environment wherein nodes can repeatedly change its geographical location.

The Local Recency and Local Frequency algorithms have two services as token circulation and event ordering. The token circulation considered by the totally ordered message delivery is achieved by continually circulating a token through all the neighbor nodes of the network. Additional mechanisms are needed depending on the desired level of reliability due to changing of the capacity of the network. In our approach, each message is circulated by the sender node, attached with a sequence number obtained from the token. An alternative approach is to store the messages in the token itself since the token visits all nodes in the coverage range, the messages will eventually reach all the neighbor nodes, the order in which messages are added to the token determining the order in which they are delivered to the nodes. The LF approach would result in large tokens (since messages are carried in the token itself). LR and LF approaches depend on the existence of a traditional wireless network.

2.4 PROPOSED WORKS

It presents local frequency distributed algorithm that causes a token to continually circulate through all the neighbor nodes of a mobile ad hoc network in a frequent of 2 milliseconds. An important application of the proposed LF algorithm is to ensure the group communication by event ordering the message delivery. Totally ordered message delivery is achieved by continually circulating a token through all the nodes of the network. The token circulates around the coverage range of the sender carrying a sequence number along with the initial packet. When a node receives the token, it assigns sequence numbers to its messages and then multicasts the messages to the group members. The sequence number carried in the token is incremented once for each message sent by the node holding the token. Since the messages are assigned globally unique sequence numbers, the total order can be achieved.
Mechanisms for finding approximations to a virtual ring that change dynamically as the topology changes and that are efficient according to certain metrics. Since token circulation around a virtual ring is a useful component of many existing group communication mechanisms for wireless networks, we will consider different ways of improving the performance of such mechanisms in mobile ad hoc networks.

2.4.1 Token Circulation Algorithm

The sender nodes transmit initial packets with the sequence number to its neighbor by checking the particular group among the several groups in a network. In distributed algorithm (Figure 2.1) that causes a token to continually circulate through all the nodes of a mobile ad hoc network. The token has circulated only to the appropriate group members around the territory; other nodes which are not in that group could not receive that initial packets. The group has been identified by its group header. The leader selection and token circulation have been discussed in section 2.5. For the multicasting, the cluster will be chosen, that was explained in chapter 5. The sequence number carried in the token is incremented once for each message sent by the node holding the token. Since the messages are assigned globally unique sequence numbers, the total order can be achieved.

2.4.2 Local Vs Global

In local algorithms, choosing the next node in that group is achieved by sending the token from the leader that is believed to be neighbors of the node having the unique identification of their leader. A global algorithm may direct the token towards any node in the network without checking their leader. It causes unnecessary packet congestion in that node as well as network.
2.4.3 Recency Vs Frequency

In a case of recency algorithms, the neighboring nodes were identified by based on how recently send the acknowledgment to their leader. By using such kind of LR algorithm in MANET, the same node may overload due to its nearest presence. In this case, the drawback was eliminated using frequency algorithm. The decision is based on how frequently the nodes have had the token. In MANET nature, the node may change its location so that visiting “next” node should be identified in a frequent manner. This variation is only relevant for global algorithms. Once the desired destination has been determined by a global algorithm, there are two possibilities:

i. The token is sent directly to the chosen destination with its unique group identification, other intermediate nodes on the route will route the token, but the message will not be processed by the intermediate nodes.

![Token circulation tree](image)

*Figure 2.1 Token circulation tree*
Alternatively, the current Leader may send the token to the next node on the route to the chosen destination effectively. The token will visit all neighbor nodes in their group (or the network) on the route to the chosen destination. In order to implement the second mechanism, the network layer must be able to provide to the application layer the identity of the next node on the route to the desired destination. When a node receives the token, it chooses the next recipient of the token using this count information, updates its own count information in the token, and sends the token to the chosen next recipient. The criteria for choosing the next token recipient and updating the count depend on the particular algorithm.

2.4.3.1 Local-Frequency

The Local-Frequency (LF) algorithm keeps track of the visited node of its territory by sending the token and its reply time. The least frequently visited nodes from a leader have predicted as the edge node. To implement this LF algorithm, the count and sequence number, for each node, were stored in the token, contains group identification the number and past token visits to that node. The Leader may not have a precise knowledge of its neighbors; occasionally the chosen node may no longer be its neighbor.

The TCP connection is used for the token delivery to protect against the potential loss of the token. This TCP contains the acknowledgment so that it is used for calculating the frequency time delay during the transmission. The TCP protocol, running on top of a multicast routing protocol for MANET, will eventually deliver the token to the intended recipient. This approach is used for all our LR algorithms. In our simulations, the AODV Routing protocol is used for routing in ad hoc networks. This proposed algorithm ensures that if there is mobility and the topology is changed, every node in the coverage area is visited infinitely often. Along
with this, the multicast communication ensures the reliable and secure communication among its groups. Each group has its own identification for providing authentic token circulation. It has been explained in chapter 4. The Figure 2.17 and 2.18 have shown the evidence for the above-mentioned algorithm combination.

2.4.3.2 Local-recency

To implement this algorithm, the count for each node, as stored in the token, contains the “time” (as defined earlier) when the node was last visited by the token. The Local-Recency (LR) algorithm is similar to LF, except that the least recently visited neighbor of the token-holder is chosen as the next recipient of the token. It ensures a round length that is never more than seven. In any static connected graph, round length of at most ‘2n’, where n is the number of nodes in the graph, can be achieved if the nodes are visited according to a spanning tree of the graph, backtracking where necessary. A similar argument to that for LF shows that there is no starvation in the case of static connected topologies. In fact, the behavior of LR is much better than that of LF on the static but not suited for the MANET. The LR algorithm attempts to improve on this round length by taking advantage of cycles to avoid the backtracking. In most topologies, the LR algorithm succeeds in improving on the ‘2n’ bound.

![Figure 2.2 Message passing between nodes](image-url)
2.4.4 Choosing the Next Token Recipient

In most of our algorithms, the next recipient is the node, among those allowed to be considered, with the smallest count value, ties being broken either arbitrarily or by some other criteria. The global algorithm uses flooding the packets to its entire neighbor for choosing the next recipient (Figure 2.2). But in Local algorithms are allowed to consider nodes that are currently on the leader’s list. The unique group number has incorporated with the token for the identification of the group nodes. The token contains one count for each node in the network; each count can potentially grow without bound although overflow is unlikely with, say, 64 bits allocated per count.

2.4.5 Event Ordering

In distributed computing there is a large number of tasks which can all be traced down to the fundamental problem of constructing a consistent global view on a distributed computation. Examples of these tasks are monitoring, break pointing, debugging, and the detection of deadlocks or global predicates in general. All of these tasks are part of the necessary effort to better understand, design, and control distributed systems. The fundamental problem of constructing a consistent global view derives from the need to issue meaningful statements about a whole distributed computation on the one hand, and the lack of a global common time base on the other. The latter issue must be overcome by ordering the events of the computation based on their mutual causal dependencies rather than a global clock. Several approaches and solutions have been given, with varying focus and background. But all of these authors based their examinations on a system model that has its distributed processes communicate solely via messages sent from one process to another.

We focus on an extension of this common model by accounting for an abstract memory to be shared by the distributed processes. The context of general consistency models and their verification has performed. Our goal is
the consolidation of all the relevant parts of the related work on the one hand, and the suggestion of a set of useful definitions, interconnections, and basic considerations on the other. This work is meant as a base to be worked upon and to be further refined in future research. We gradually extend these concepts by introducing a shared memory and the resulting problems, and by giving and refining basic definitions. We also prove a theorem that states necessary and sufficient conditions for a consistent global order.

The LF algorithm circulates the initial packets with a sequence number. The token circulation was to ignore the network topology and choose an arbitrary order to visit the nodes. A territory consisting of 8 nodes are visited from node 0, in this case, node 0 acted as a leader. By the same manner the token has circulated with the count to the first level of neighbor 2, second level 1,4,3,6,7 and 8, the token takes several hop in this case, although the length of the round is 4 (which is optimal), message overhead is large since each node visit requires the token to take several hops). The visited nodes from leader 0 in the order 2, 1, 3, 4, 6, 7, 8 still results in the optimal length, but lower message overhead.

The token has been checked its group number before transmission. Node 5 has identified as a separate group which was identified in round 3. Due to this filtering mechanism, the unauthenticated nodes were not able to receive the token, unnecessary time delay and power loss were reduced. By the same time, the different territory node can exempt from the congestion and it was free for their territory token circulation. If the existing leader may move away from their group and joins to the new group, then the node updates its identification by the new region but keeping the old group id for future. The leader election algorithm is used for finding the new leader. This kind of searching, sending, circulating and electing process was continuously performed. Due to this, the power was drained in a drastic manner.
The Figure 2.3 suggests the order of the nodes were visited for and used to determine the network topology information while visiting. However, knowledge of the network topology, particularly in mobile environments, is expensive to achieve. Thus, the latter visit order should be preferred. However, if the visit order is chosen without taking the topology into account, in general, the algorithm will not typically choose the best possible order of visits.

Figure 2.3 Token circulation
The initial packets with group key are used for communication service, or alternatively, this information may be obtained by the network layer and made available to the group communication service via a system call. Therefore, the token circulation algorithms use only local frequency information and also consider an algorithm that uses dynamic topology information. The simple approach for keeping track of neighbors in a network is forwarding “hello” messages to its entire neighbor node periodically. The time between the periodic hello packets referred as the hello interval.

### 2.4.6 Updating the Counts

The token carries a direction with the count for each node. The recency and frequency algorithms update the count information in the token differently, as described next. For recency algorithms, the count for the node (as stored in the token) represents the last “time” when the node was visited. “Time” in this case represents the total number of visits made by the token. The token carries a time variable, in addition to the counts. The time variable is initialized to 0 and then incremented by 1 each time the token visits a node. Assume that on a particular token visit to node \( v \), the time variable in the token is incremented to 15. Then, the count for node \( v \) (as stored in the token) will be updated to be 15 as well.

For frequency algorithms, the count for the node is the number of times the token has visited node. Each node maintains a local variable to store the number of token visits made to that node. Each time the token visits a node, the node increments its local counter by 1, and the new value is also stored in the Node’s count in the token. Thus, the token, at all times, contains the number of visits made by the token to each node.
2.4.7 Election Algorithm

The coordinator election problem is to choose a process from among a group of processes on different processors in a distributed system to act as the central coordinator. An election algorithm is an algorithm for solving the coordinator election problem. By the nature of the coordinator election problem, any election algorithm must be a distributed algorithm.

2.5 IMPLEMENTATION

The performance evaluation results for the LR and LF algorithms are discussed below. The performance evaluation is done with the ns-2 simulator with CMU extensions. Here, the network consisting of 11 nodes has been considered for our implementation. Initially, node 0 is a leader. The transmission range of each node is 250m and the bandwidth of the channel is 2Mbps. To model mobility of the nodes, we used the random waypoint mobility model. In node mobility scenario generated using this model, the 10 nodes are initially placed in randomly chosen positions in a 500m x 400m box. Then, the nodes follow randomly chosen paths. For our experiments, we used node speeds of 6, 12, 18, and 24 m/s and the duration of the simulation was varied inversely with the speed, with the duration for the slowest speed (6m/s) being 50 seconds. Each algorithm runs as an application on top of TCP, the AODV protocol, and IEEE 802.11b MAC. By using TCP as the transport protocol, we ensure that the token does not get lost due to route failures or transmission errors.

To facilitate implementation of the LR and LF algorithms, we augmented AODV such that the token circulation algorithm can obtain the next node on the route to any given destination and it does not need to make use of hello messages for this purpose. The “hello” protocol to maintain neighborhood information this protocol is used for LF, LR one of the
simulation runs for the Iterative Search algorithm. Specifically, the metrics are the average time in seconds per round, the average number of bytes transmitted per round (including any hello packets, TCP packets, and medium access control packets), and the average number of nodes visited per round. The hello interval was varied as explained later. In our performance evaluation, we measured average values of the metrics. The following parameters were varied Hello interval. In simulations, hello interval values of 0.1 and 0.3 seconds are used. As described earlier, some of our algorithms find it useful to know the neighbors of the node that holds the token. An issue of interest to us is the frequency with which the hello messages are transmitted. Of course, since the nodes are mobile, it is not possible to maintain perfect knowledge of the neighborhood. The accuracy of this information may affect the overhead of the token circulation mechanisms. Greater frequency results in greater accuracy in the neighborhood information, but also the greater overhead of the hello messages. The issue of hello frequency has been previously studied in the context of unicast routing in ad hoc networks; however, here we consider the impact of hello frequency (or hello interval) on the overhead of token circulation algorithms.

2.5.1 Pseudo Code

We present the pseudo code of the proposed work here

Step1 when node x receives token from node y :

Step2 neighborCount [y] := token.count

Step3 if(token. visited[x] = false) then

Step4 token.visited[x] := true

Step5 if (x ≠ y) then prevleader:= curLeader; curLeader: = y; endif

Step6 endif
Step 7: next := getNext()
Step 8: neighborCount[next] := token.count
Step 9: send token to next
Step 10: function getNext() returns node id
Step 11: N := set of ids of all neighbors of x
Step 12: UV := \{z : token.visited[z] = false\}
Step 13: UV N := N \land UV // unvisited neighbors
Step 14: if (|UV|) = 0 then // all nodes are visited
Step 15: token.visited[z] := false, 0 ≤ z < n
Step 16: token.counting := false
Step 17: token.count := 0
Step 18: return x
Step 19: else if (|UV N| = 0) then // backtrack
Step 20: token.counting := true
Step 21: token.count := 1 + max\{token.count\} U \{neighborCount[z]: z \in N\}
Step 22: return curSender
Step 23: else // there is an unvisited neighbor
Step 24: token.counting := false
Step 25: token.count := 0
Step 26: return any such k
Step 27: else
Step 28: m := min\{neighborCount[z]: z \in UVN\}
Step 29: if prevLeader \in S then return prevLeader
Step30 else return any $z \in S$ endif
Step31 endif
Step32 endif

2.5.2 Simulation Model

In this case, we generated many connected random graph topologies for the 11 nodes and simulated the various algorithms. Since the topology is dynamic, the routes, once determined using AODV, do not break during the simulation. These are plots, for each algorithm, nodes visited, the number of bytes sent, and the amount of time elapsed per round. For the number of nodes visited per round, the LF algorithm performs the best than the LR; this is of course by definition, since we are only counting the number of visited nodes and not the number of nodes that relay the token. But the GF and GR algorithms pay a cost in terms of bytes and time for having the perfect round length. Good performance in terms of round length is exhibited by the Iterative Search algorithm, which converges to the optimal round length after some time and by the LF algorithm, which within one round converges to close to the optimal round length. In Section, we mentioned that the GR algorithm had the unfortunate property that the round length can increase without bound in certain topologies. Our simulation results indicate that this property of the LF algorithm occurs in many graphs rather than on a small set of graphs.

For the simulation of dynamic topologies have two sets of plots

i. First, we vary the speed of the mobility inside the network nodes to find the average amount of time. Then we calculate the bytes size and the average number of nodes visited per round for all the scenarios with the different speeds.
ii. Second, we vary the hello intervals and find the average amount of time, an average number of bytes, sequence number, count and the average number of nodes visited per round for all the scenarios with the different hello intervals. Recall that the duration of the simulation was varied inversely with the speed with the duration for the slowest speed (5 m/s) being 50 seconds. When we simulated mobility, the LF algorithm continues to perform well in all situations, similar to the dynamic topology cases. In order to investigate the packet delivery, reliability and overhead performance of LF, we used the following metrics: throughput, reliable-throughput, receive-overhead and send-overhead.

Throughput, in other words, Token delivery ratio, is calculated as the fraction of the total numbers of data packets delivered to the receivers with the actual number of data packets to be delivered. It indicates to what extent the protocol succeeds the delivery of the packets originated. Reliable-throughput is the fraction of the data packets delivered to all receivers over the total number of data packets supposed to be delivered. That is, a packet should be delivered to all members of a multicast group in order to be counted in reliable throughput. It shows how successfully the protocol can deliver the data to all receivers.

Receive-overhead is the ratio of bytes belonging to control packets and duplicate data packets received by a node to the bytes of data were delivered. That is, all the bytes received, except the bytes of data packets received for the first time, are counted in the receive-overhead portion of the protocol. Receive-overhead illustrates the network load that a node exposes to deliver one byte of data. In ad-hoc networks, receive-overhead may be misleading since the nodes in such networks can hardly receive the packets.
This situation triggers a drop in the network load noticed by the nodes and so causes a receive-overhead lower than it actually. Send-overhead is the ratio of total bytes sent by a node to the bytes of data it delivered. However, it may be inadequate to show the actual network load in broadcast transmission since multiple receivers may receive the packet. Thus, it should be taken into account together with receive-overhead.

2.6 SIMULATION RESULTS

The simulation graphs (Figure 2.17 and 2.18) show the plots of the average time versus speed and average number of nodes versus speed for the LR, LF algorithms. For the time the algorithms from best to worst are ordered: LF and LR with hello packets for MANET using AODV routing protocol. More specifically, the average time per round of LF and LR increases very slowly with speed. Similarly, the average number of nodes per round of LR and LF increases very slowly with speed, with hello. For round length, the ranking is consistent except for the fact that LR and LF by definition are optimal.

We present two test cases in detail. The detailed scenario of each test case and its outcome is given below.

Test case 1

i. A system consisting of 11 nodes (500m x 400m box).

ii. The transmission range of each node is 250 m and the bandwidth of the channel is 2Mbps.

iii. Initially, node 0 designated as a coordinator. It displayed in text message bar.
The following figures from 2.4 to 2.16 show the coordinator election and token circulation and new leader election among the experimental nods. The graphs (Figure 2.17 and 2.18) show the comparison of Local Recency and Local Frequency algorithms. Figure 2.4 shows the initial set up of our simulation. In this node 0 is considered as a leader. It has the authentication for initiate token circulation to its neighbor. Figure 2.5 shows the neighbor token circulation with the count value.

The Figure 2.6 shows checking of the group members for forwarding the packets. The node 4 has been identified as its group, the other nodes 1 is not belonging to its group and node 2 is moved away. It indicates that our simulation has experimented with dynamic topology.

Figure 2.4 Coordinator node
Figure 2.5 Token distribution

Figure 2.6 Message passing
Figure 2.7  Distributing the tokens to neighbour

Figure 2.8  Token received by neighboring nodes

Again node 4 forwards the packets to 6, 3 Likewise Figures 2.7 to 2.9 shown the packet circulation to our groups and find the destination from leader node 0.
The above Figure 2.10 shows the entire token circulation environment. The other nodes were belonging to separate clusters inside the same network.
Test case 2

If the leader moves away or crashed after delivering the message to the intermediate node, then a new node has to elected as leader

i. The message received node sending election message to its neighbor nodes

ii. After getting the reply message from any one of its neighbor nodes, the replied node elected as a new coordinator.

iii. If else, presently message containing node elected as a coordinator by itself.

Figure 2.11  Leader moves away from the territory

As we said earlier we are considering the MANET environment with dynamic topology. The node 0 leaves from the range that shown in Figure 2.11.
Figure 2.12    Conducting election

Figure 2.12 and 2.13 shows that the node 4 initiating the election algorithm to find out a new leader. Its sends the election message to its neighbor nodes and getting the reply from nodes 1, 5, 6. The node 5 belongs to a different group so the reply from the node 5 discarded. Compared the frequency level of the nodes 1, 4, 6 node 4 has participated the last transmission and it has higher frequency level than the other two. So the node 4 has been elected as a new leader or coordinator for their group nodes. This was experimented by using the LF and LR algorithms. But comparatively, the LF provides very fast communication and behaves as ideal for mobility.
Figure 2.13  Election message received by the nodes

Figure 2.14, 2.15 shows that the new leader node 4 sends a message to its neighbor nodes and reaches the destination. The node 5 belongs to a different group in the previous transmission (Figure 2.14) has entered into our group due to the roaming nature. So the message has reached the destination via nodes 5, 8, 10. It was shown in the Figure 2.16.

Figure 2.14  Elected a new node as a coordinator
Figure 2.15  Token distributed by the new leader

Figure 2.16  Message sending from new leader
Figure 2.17  Average time Vs speed

Figure 2.18  Number of nodes Vs speed
2.7 CONCLUSION

The section concludes with the advantages of leader election with recency and frequency algorithms. If the existing leader has crashed or moves away from the territory means the proposed leader election algorithm has run and elected the new leader with the use of Local Recency and local Frequency algorithms. The LF evidenced that it has better performances than LR during transmission. The next chapter discuss about the different types of routing which have helped for efficient routing for MANET.