Firefly based Optimized Routing over MANETs

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Technology

in

Computer Science and Applications

Submitted By
Rahul Gupta
(Roll No. 601303023)

Under the supervision of:
Dr. Rajesh Kumar
Associate Professor, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2015

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Firefly Algorithm based Optimized Routing over MANETs*", in partial fulfilment of the requirements for the award of degree of Master of Technology in *Computer Science and Applications* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Rajesh Kumar* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Rahul
(Rahul Gupta)

601303023

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Rajesh Kumar)

Associate Professor, CSED

Countersigned by

(Dr. Deepak Garg)

Head

Computer Science and Engineering Department

Thapar University

Patiala

(Dr. S. S. Bhatia)

Dean (Academic Affairs)

Thapar University

Patiala

i

ACKNOWLEDGEMENT

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life.

This work would not have been possible without the encouragement and able guidance of my supervisor **Dr. Rajesh Kumar**. I thank my supervisor for his time, patience, discussions and valuable comments. Their enthusiasm and optimism made this experience both rewarding and enjoyable.

I am also thankful to **Dr. Deepak Garg,** Head, Computer Science and Engineering Department; **Dr. Sanmeet kaur**, coordinator, M.Tech. CSA, for their constant support and encouragement.

I am also thankful to **Vishal Sharma** for discussing his valuable ideas and comments with us.

I will be failing in my duty if I don't express my gratitude to **Dr. S. S. Bhatia**, Professor and Dean of Academic Affairs of University, for making provisions of infrastructure such as library facilities, immensely useful for the learners to equip themselves with the latest field.

I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

Last but not least, I would like to thank my parents for their wonderful love and encouragement, without their blessings none of this would have been possible. I would also like to thank my close friends for their constant support.

ABSTRACT

Mobile Ad Hoc Networks (MANETS) are the infrastructure less network that are very dynamic in nature. The main problem with such network is mobility of the nodes; the nodes are free to roam arbitrarily, thus the topology changes time to time, and route acquisition in such a network is always a tedious task. MANETs are dynamic multi hop wireless network which is set-up by a collection of mobile nodes on a shared medium. Routing is transfer of information across the network from a source to a destination. For an efficient network formation, mobile nodes should be capable enough to utilize the available query resources optimally. When it comes to MANETs, the complexity increases due to various characteristics like dynamic topology, limited resources and energy etc.

In this dissertation, we have proposed an algorithm based on firefly algorithm (an implementation of swarm intelligence). The algorithm captures the nature of fireflies i.e. how they do the task individually and how they coordinate with each other to create a system. The proposed algorithm is compared with the state-of-the art routing algorithm and it improves MANETs routing performance in terms of packet delivery ratio, packet loss, throughput and delay. The proposed algorithm applies the idea of firefly algorithm on the mobile ad-hoc network that enhances the performance of routing by efficient packets transfer from source node to destination node via intermediate nodes. Results showed that the proposed algorithm performs better than AOMDV routing protocol in terms of higher packet delivery ratio, minimum packet loss, higher throughput and better end-to-end delay.

LIST OF FIGURES

LIST OF SNAPSHOTS

LIST OF TABLES

TABLE OF CONTENT

Chapter 1
Introduction

1.1. Overview of MANETs

The collection of wireless mobile nodes is called Mobile Ad-Hoc Networks (MANETs) [1] that make a momentary network not using any centralized access points or centralized administration. MANETs comprise of mobile nodes that have capabilities of communicating with each other using packet radios over a shared wireless medium. It should be easy for nodes in the network to sense and discover nearby nodes [2]. But due to small range of wireless network, it needs multiple network hops to exchange data and information with other nodes over the network. This network is useful because of having a lot of advantages in terms of cost and flexibility as compare to wired network [3]. The main application areas of MANETs are data collection, seismic activities, medical applications, military applications, rescue operations, wearable devices and in other places where pre-installed infrastructure is not possible.

In a MANET, all the participants (nodes) are mobile and use only wireless communication links to send and receive data packets. Self-organisation, configuration and maintenance are built into the network, in the form of network protocols which need to be highly dynamic and operate in a fully distributed way. They also need to be adaptive to any topological changes and robust in order to face unreliable links between transmitting nodes. Finally, due to the network limited resources, protocols need to provide cheap solutions in terms of energy consumption, bandwidth and computing power.

The two major challenges for the above systems are node mobility and limited energy resources. The impact of node mobility on a routing protocol's performance is an established research problem and can mainly be seen as the product of frequent link breaks, energy consumption and changes to the node density within the topology. The higher the density of the network, the greater the load within it. Throughput is affected as more control overhead is broadcast to the network and the greater the node density, the greater the packet loss rate. Clustering is also a frequent problem with

node mobility where traffic is routed through the nodes at the group edges, causing bottlenecks and load congestion.

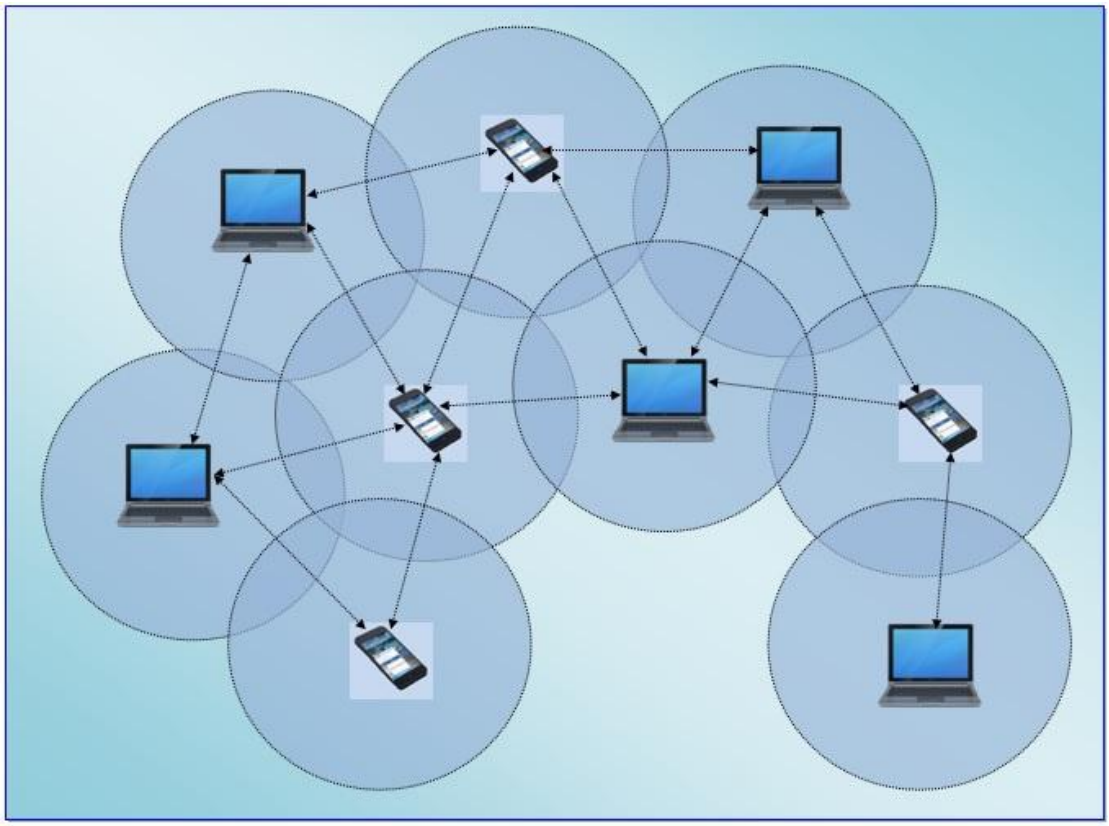


Figure 1.1. MANETs Overview

Nodes consume energy while sending, receiving and even discarding packets. Studies have shown that memory allocation at the mobile node consumes a high level of power. Route discovery, routing maintenance and self-organisation of topology require extra control packets, thus requires more energy. Power consumption in wireless ad hoc networks [4] is directly proportional to route length.

1.2. Characteristics of MANETs

- *Distributed operation:* The control of the operation in MANETs is not operated by a single central nodes network. The control is distributed among several nodes in the network. The communication among these nodes is the backbone for all the network operation like routing, security etc.

- *Multi hop routing :* In MANETs all the mobile nodes are identified by their ranges and when a nodes needs to send the information to a node which is out of

its range, the information in the form of packets is transferred by intermediate nodes that share communication range with each other as shown in Figure 1.2.

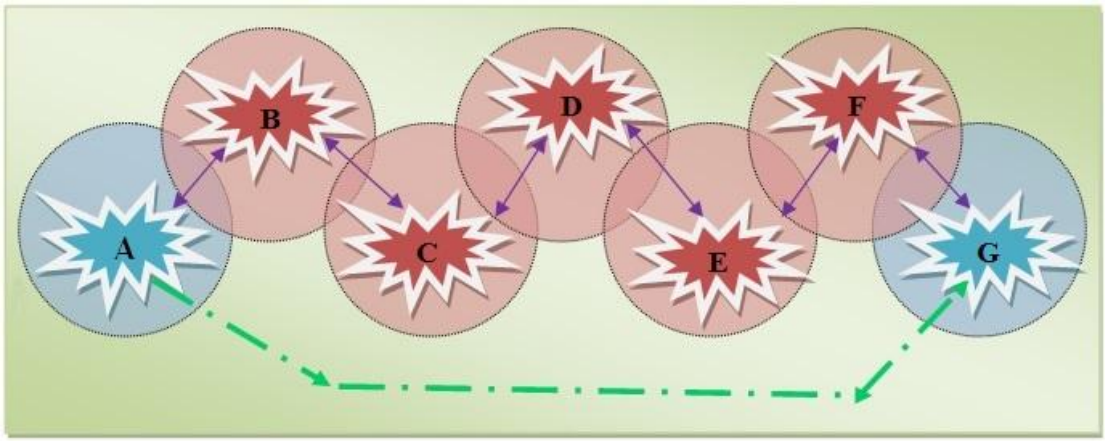


Figure 1.2. Multi Hop Routing: Node A Send Information to Node G via Intermediate Nodes Forming a Sequence with Common Communication Range

- *Dynamic Topology:* MANETs are highly dynamic in nature as the mobile nodes are free to move at random with uneven speed, i.e. why the network topology in MANETS is volatile and changes with time.

- *Light-weight terminals:* The mobile nodes in MANETs are generally lightly weighted, that means they requires very few CPU capability, power storage and memory size.

1.3. Issues with MANETs

MANETs should take care of stringent time and reliability-sensitive service requirements that multimedia applications need the most. MANETs should fulfil QoS requirements. Earlier techniques cannot guarantee today's requirements. So many solutions are proposed to fulfil quality metrics. It is a big challenge to provide better quality of service because of the following:

- *Node Mobility:* Nodes in MANETs are mobile; it makes the topology dynamic in nature. It means a limited life is there for topology information and for sending packets from source to sink therefore its frequent updating is necessary. This updating increases the routing overhead. It affects the end to end delay and packet loss is also increased.

- *Lack of Central Control:* The advantage of MANETs is that it is deployed without any planning and can be changed dynamically. But, it is difficult to have any centralized control. So the control activities are distributed all over the nodes that require a lot of information exchange, which adds overhead to the routing.

There are already many routing protocols like AODV, TORA, DSR etc, but still there is need of improvement. With increase in the growth of internet, the demand for quality of service is also increasing.

1.4. Classification of MANETs Routing Protocol

Routing protocols for mobile ad hoc networks are traditionally classified into several types (proactive, reactive, or hybrid) as shown in Figure 1.3, based on their routing approach and mechanisms [5,6,7].

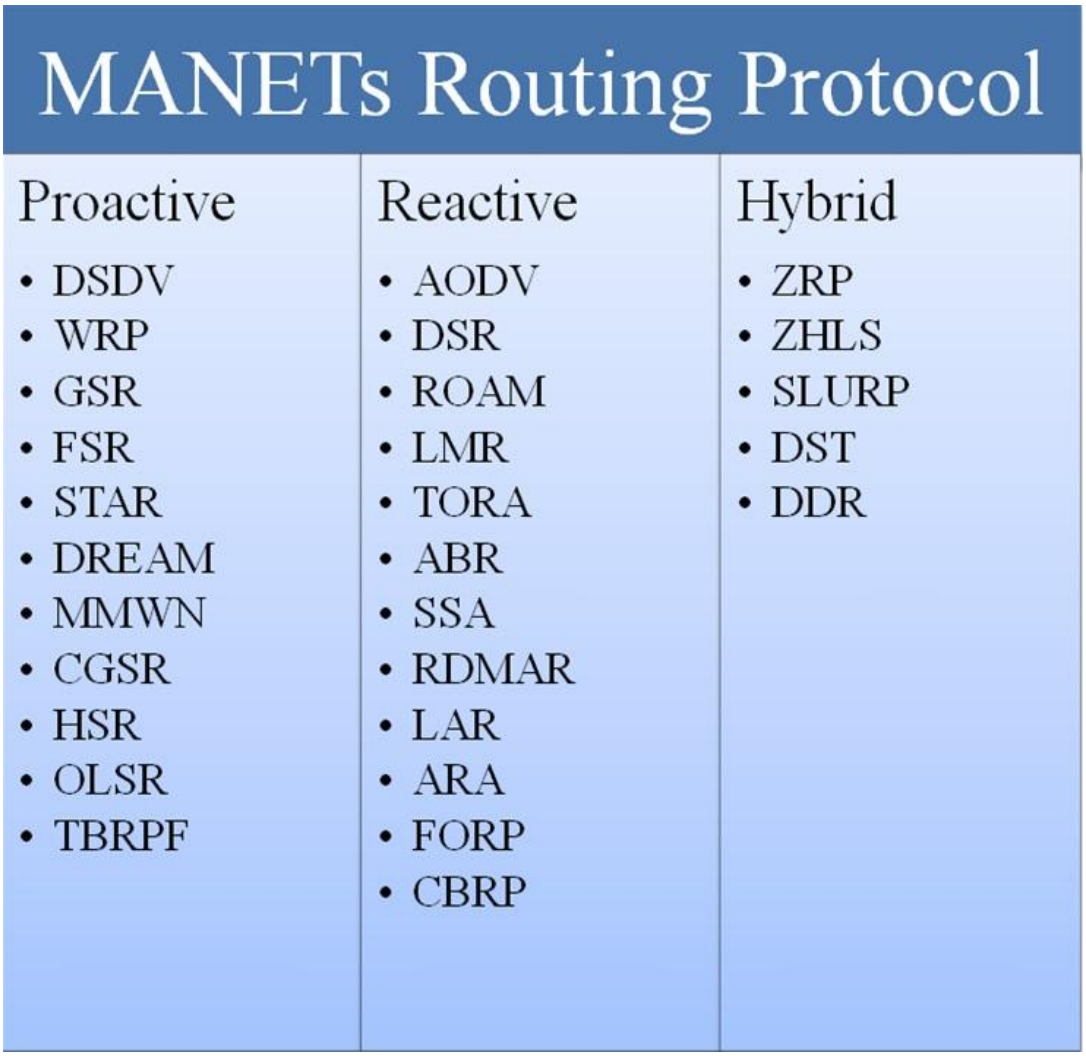


Figure 1.3. Classification of MANETs

In proactive or table-driven routing protocols, the network topology information is maintained within each node in the form of routing tables, which are periodically exchanged in order to keep the information up-to-date. The routing table are also updated whenever network topology changes. Destination Sequenced Distance-Vector (DSDV), a proactive routing protocol, is considered to be the de facto standard in the area of proactive routing in MANETs. Although proactive routing is often seen as not the best approach for wireless ad hoc networks, especially under large number of nodes and high mobility rates. DSDV is thoroughly studied and used in protocol comparisons in the literature. The comparison of various proactive routing protocols is shown in Table 1.1.

Table 1.1. Proactive Routing Protocols of MANETs

PROTOCOL	FULL FORM	FREQUENCY OF UPDATES	CHARACTERISTIC FEATURE	ADVANTAGE	DISADVANTAGE
DSDV	Destination-sequenced distance vector	Periodic and as required	Loop free	Loop free	High overhead
WRP	Wireless routing protocol	Periodic	Loop freedom using predecessor info	Loop free	Memory overhead
GSR	Global state routing	Periodic and local	Localised updates	Localized updates	High memory overhead
FSR	Fisheye state routing	Periodic and local	Controlled frequency of updates	Reduces CO	High memory overhead, reduces accuracy
STAR	Source-tree adaptive routing	Conditional	Employees LORA and/or ORA. Minimize CO	Low CO and MO	High MO and processing overhead
DREAM	Distance routing effect algorithm for mobility	Mobility based	Controlled rate of updates by mobility and distance	Low CO	Requires GPS
MMWN	Multimedia support in mobile wireless network	Conditional	LORA and minimized CO	Low CO	Mobility management and cluster maintenance
CGSR	Cluster-head gateway switch routing	Periodic	Cluster-heads exchange routing information	Reduced CO	Cluster formation and maintenance
HSR	Hierarchical state routing	Periodic, within each subnet	Low CO and Hierarchical structure	Low CO	Location management
OLSR	Optimized link state routing	Periodic	Reduce CO using MPR	Reduced CO and contention	2-hop neighbour knowledge required
TBRPF	Topology broadcast reverse path forwarding	Periodic and differential	Broadcasting topology updates over a spanning tree	Low CO	High MO

CO = control overhead; MO = memory overhead; LORA = least overhead routing approach; ORA = optimum routing approach; MPR = multipoint relaying.

Unlike proactive protocols, reactive (also called on-demand) routing protocols for MANETs request for the route discovery only when needed. They do not store the

path unless it is required. The route request is broadcasted to the neighbour of the node and continue in the same way. A route reply is returned either by the destination or by an intermediate node with an available route. Ad hoc On-Demand Distance-Vector protocol (AODV), is a reactive routing protocol which is thoroughly studied in the literature.

Table 1.2. Reactive Routing Protocols of MANETs

PROTOCOL	FULL FORM	ROUTE METRIC METHOD	ROUTE RECONFIGURATION STRATEGY	ADVANTAGE	DISADVANTAGE
AODV	Ad hoc on demand distance vector routing	Freshest & SP	Erase route then SN or local route repair	Adaptable to highly dynamic topologies	Scalability problems, large delays, hello messages
DSR	Dynamic source routing	SP, or next available in RC	Erase route the SN	Multiple routes, Promiscuous overhearing	Scalability problems due to source routing and flooding, large delays
ROAM	Routing on-demand acyclic multi-path	SP	Erase route	Elimination of search-to-infinity problem	Large CO in highly mobile environments
LMR	Light-weight mobile routing	SP, or next available	Link reversal & Route repair	Multiple routes	Temporary routing loops
TORA	Temporally ordered routing algorithm	SP, or next available	Link reversal & Route repair	Multiple routes	Temporary routing loops
SSA	Signal stability adaptive	Strongest signal strength & stability	Erase route then SN	Route stability	Scalability problems, large delays during route failure and reconstruction
RDMAR	Relative distance micro-discovery ad hoc routing	Shortest relative distance or SP	Erase route then SN	Localised route discovery	Flooding used if there is no prior communication between nodes
LAR	Location-aided routing	SP	Erase route then SN	Localised route discovery	Based on source routing, flooding is used if no location information is available
ARA	Ant-colony based routing algorithm	SP	Use alternate route or back track until a route is found	Low overhead, small control packet size	Flooding based route discovery process
FORP	Flow oriented routing protocol	RET & stability	A Flow_HANDOFF used to use alternate route	Employees a route failure minimisation technique	Flooding based route discovery process
CBRP	Cluster-based routing protocol	First available route (first fit)	Erase route then SN & local route repair	Only cluster-heads exchange routing information	Cluster maintenance, temporary loops

RC = route cache; RET = route expiration time; SP = shortest path; SN = source notification; LBQ = localised broadcast query.

AODV's performance is generally considered to set the benchmark standards in evaluating routing protocols for MANETs, as the protocol participates in most of the

existing work in this research area. This is also due to the current experimental status of its draft by the IETF MANET group, which sets AODV under investigation for standardization. The comparison of various reactive routing protocols is shown in Table 1.2.

Ad hoc On-Demand Multi-path Distance-Vector (AOMDV) is a multi-path extension to AODV. Dynamic Source Routing (DSR), also a reactive routing protocol, has gained considerable attention because of its ability to dramatically reduce control overhead, even under high rates of node mobility. DSR is similar to AODV in terms of route discovery, however it uses source routing in order to forward packets. Source routing reduces the computation complexity of deciding the next hop, as well as the amount of storage required to keep routing information, at each intermediate node.

Hybrid routing protocols are a new generation protocol, which are both proactive and reactive in nature. In this kind of protocol, nodes with close proximity to works together to reduce the route discovery overheads by creating some sort of backbone. This procedure helps in increasing scalability of the network. This is mostly achieved by proactively maintaining routes to nearby nodes and reactively determining routes to far away nodes using a route discovery strategy. Most hybrid protocols proposed till date are zone-based, which means that the network is partitioned or seen as a number of zones by each node. The comparison of various hybrid routing protocols is shown in Table 1.3.

Table 1.3. Hybrid Routing Protocols of MANETs

PROTOCOL	FULL FORM	ROUTE METRIC METHOD	ROUTE RECONFIGURATION STRATEGY	ADVANTAGE	DISADVANTAGE
ZRP	Zone routing protocol	SP	Route repair at point of failure and SN	Reduce retransmissions	Overlapping zones
ZHLS	Zone -based hierarchical link state	SP or next available virtual link	Location request	Reduction of SPF, low CO	Static zone map required
SLURP	Scalable location update routing protocol	MFR for interzone forwarding. DSR for intrazone routing	SN, then location discovery	Location discovery using home regions	Static zone map required
DST	Distributed spanning trees based routing protocol	Forwarding using the tree neighbours and the bridges using shuttling	Holding time or shuttling	Reduce retransmissions	Root node
DDR	Distributed dynamic routing	Stable routing	SN, then source initiates a new path discovery	No zone map or zone coordinator	Preferred neighbours may become bottlenecks

SP = shortest path; SN = source notification

1.5. Protocol Considered

1.5.1. Ad Hoc On Demand Distance Vector Routing Protocol (AODV)

AODV routing protocol [8, 9] is a self starting routing protocol. Each mobile host operates as a specialized router. In this routing algorithm, routes are obtained as needed i.e. on demand. The routes are acquainted via little or no periodic advertisement, this makes AODV perform better as routing overhead is reduced. The routing algorithm is suitable for self starting network. This algorithm also implements the logic for loop-free routes even when the links appeared to be broken. The one drawback of MANETs is routing in huge population of mobile nodes, but this algorithm scales well to large population of nodes. The AODV is a combination of DSDV and DSR, from the DSDV it uses the periodic beaconing and sequence number procedure and the route discovery procedure used in this algorithm is taken from DSR. The problem with DSDV is that scales itself only to small population of nodes as routing overhead is high because of the global dissemination of connectivity, i.e. control messages grow with network as $O(n^2)$. AODV routing algorithm also takes small steps for route acquisition as the need for periodic broadcast of route advertisement is no longer needed because the routes are needed on demand. The algorithm uses limited range broadcast media such as those utilized by infrared and radio frequency wireless communication adapters. The benefit of using such media lies with mobile nodes, that they receive the broadcast yet they can't detect each other. The algorithm works on wireless as well as wired medium.

When a mobile needs to route a packet to a destination, it initiates the process of route discovery. In Route discovery a packet called Route Request (RREQ) is created and broadcasted on the network. This packet marks the destination as goal and waits for another packet called ROUTE Reply (RREP). An intermediate node receiving the RREQ packet, first establishes a reverse path (Figure 1.4) towards the source using the nodes in the backtrack path of nodes in RREQ. If a suitable route to the destination is found to be available at the intermediate nodes, then the intermediate node generates a RREP packet, else the RREQ is broadcasted further till the path to the destination or destination is not found.

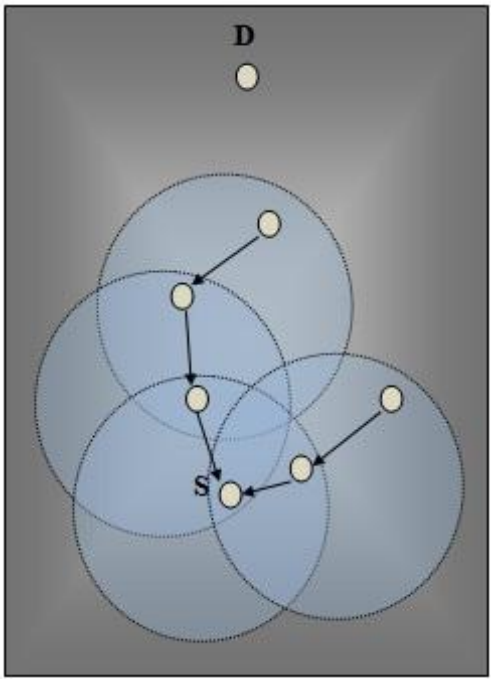


Figure 1.4. Reverse Path Formation

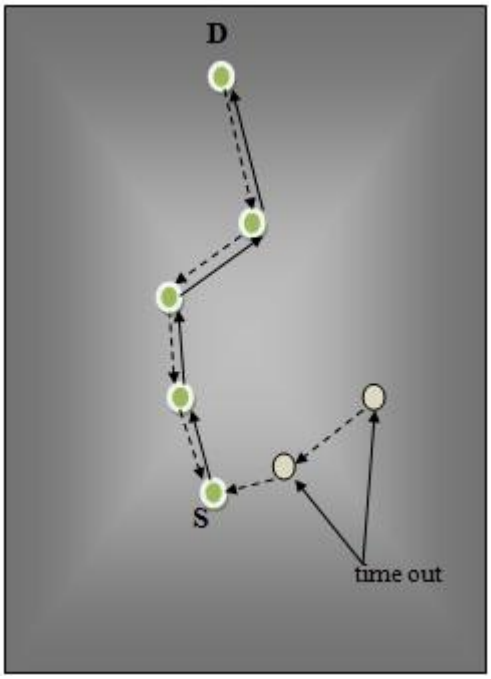


Figure 1.5. Forward Path Formation

Duplicate copies of RREQ are discarded using the sequence number concept. When the destination receives a RREQ, it generates a RREP and route it back to the source which in actual is the reverse of RREQ. As the RREP proceeds towards the source, a forward path (Figure 1.5) to the destination is established. Route maintenance is established using Route Error (RERR) packets.

1.5.2. Ad Hoc On Demand Multicast Distance Vector Routing Protocol (AOMDV)

AOMDV [10] is the advance form of AODV for computing multiple disjoint loop free paths in a route discovery. In this, each node has a unique number called unique identifier like IP address. All the links in MANETs are bidirectional in nature. AOMDV can be applied even in the presence of unidirectional links with additional techniques to help discover bidirectional paths. AOMDV possesses similar characteristics to AODV. The main concept here is distance between two nodes and uses hop-by-hop routing approach. Same as in AODV, AOMDV too calculates distant path on demand using route discovery method. But there is one thing which differentiates AOMDV from AODV is the route discovery. In AODV, if a source wants to make a connection with the destination node then it can be done by broadcasting a route request packet to the destination. Duplicate requests are detected

by a unique sequence number. The halfway nodes which are receiving request packets firstly create a reverse connection if it founds a path to the destination node. If the route is valid then the intermediate node generates a route reply to the previous node otherwise it discards the message. The node then updates its routing table and sends the RREP to the previous node only if a RREP has a larger destination sequence number or if the route found is shortest. But in AOMDV, the RREQ is broadcasted from the source to the destination making multiple reverse paths both at intermediate nodes as well as the destination. Multiple RREPs traverse these reverse paths back to the source forming multiple forward paths to the destination [33]. The core concept of the AOMDV protocol is to find multiple paths from source to destination. It ensures that the multiple paths which are discovered should be loop disjoint and free. Multiple routes are discovered by multipath routing protocol during one route discovery procedure. When the primary route fails, the multiple paths are used for back up recovery. The paths should be found efficiently using flood-based route discovery. Route update rules in AOMDV are applied on each node and play a key role for maintenance of loop-freedom and disjoint properties. AOMDV is more useful as compared to other protocols because AOMDV mostly relies on the routing information that is already available in the underlying AODV protocol, so there is very limited overhead for discovering multiple paths. In general, it does not employ any special control packets. But additional overhead as compare to that of AODV is because of extra RREPs and RERRs for discovering multipath and maintaining along with a few extra fields in routing control. The key difference between the two algorithms is AODV is single-cast and AOMDV is multicast, i.e. the AODV maintains only one path to the destination from the source while AOMDV maintains multiple path for the same. This concept of multiple path makes routing overhead very less as compared to AODV, route discovery, which is considered as a very time consuming and complex process, is needed only when no single route to the destination is available. The main goal of the algorithm is to perform efficiently in terms of fault tolerance in efficient and faster routing, here efficient routing concentrates on recovery from route failures in network. It achieves the goal by evaluating multiple path to the destination which are loop-free and disjoint paths. The algorithm for AOMDV is solely based on AODV routing algorithm.

- *Loop free paths:* Loops are formed frequently in routing algorithms which are needed to be taken care of. Consider a scenario as explained in Figure 1.6, where a node 0 broadcasts the RREQ packet to its neighbour and the neighbour 1 doesn't have a path to the destination, so it broadcast the RREQ to its neighbour. As 0 is also 1's neighbour, the RREQ will be forwarded to 1 and thus forming the cycle. So to overcome this problem, the concept of visited node is used, as used in Breadth First Search, AOMDV accepts and maintains multiple paths to the destination.

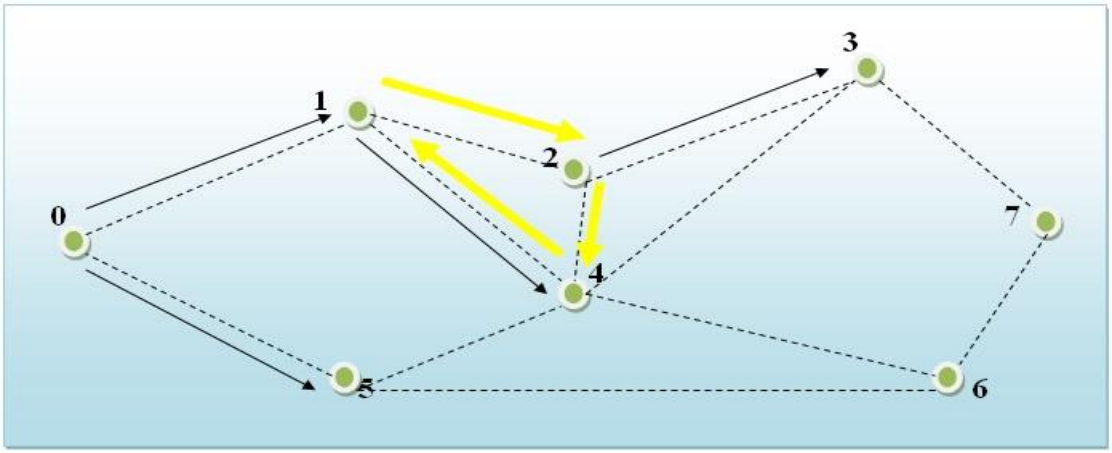


Figure 1.6. Loop Formation: Node 1,2,4 Because of Broadcasting, Creates a Loop.

- *Disjoint paths:* The paths can be made disjoint by making node disjoint and link disjoint. In node disjoint, except the source and destination, no other nodes can be common in the path and in link disjoint, no single link can be common but nodes need not be disjoint. In node disjoint, because of more restrictions, lesser number of disjoint paths are obtained, but it guarantees links fail independently. Link disjoint performs better until the links does not fail independently, this happens rarely when nodes go out of each other range because of dynamic nature of mobile nodes of MANETs.

As multiple paths are available for each node to the destination, then there occurs a problem in route discovery, i.e. which route hop count to broadcast to the neighbours. In AOMDV, for the available routes, it broadcast the route with maximum of hop count and it termed it as advertised_hop_count.

1.6. Preface to Swarm Intelligence

Swarm Intelligence (SI) is a class of evolutionary algorithms, which had proven itself as a very superior algorithm because of its nature to reduce the complexity of the task. SI is nothing but the cooperative behaviour of the natural or artificial self-organized decentralized system [11, 12]. The concept of SI is basically applied on artificial system.

SI is that class of evolutionary algorithms which is recognized for its collective group behaviour. SI typically consists of a group of tiny capable individual, who perform their task independent of each other and their environment, but the approach that makes SI so dominant is the cooperation among these individual. These individual are known as agents which obeys simple rules and are individually not capable of performing a huge task by themselves, but they can perform the big task efficiently by their group behaviour. There is no global structure given by SI that states anything about the centralized control on how these agents behave, communicate with each other. The whole effectiveness of SI depends entirely on the interaction or communication between these agents, which, in turn, leads to the globally effective intelligent system. The individuals are not aware of their capability to perform complex task via their intelligent interactions with each other. The principle of SI states that "SI should be multi-agent system that has self-organized behaviour and shows intelligent behaviour". Examples of SI includes ant colony, bird flocking, animal herding, bacterial growth and fish schooling, etc. Some applications of SI are shown in Figure 1.7.

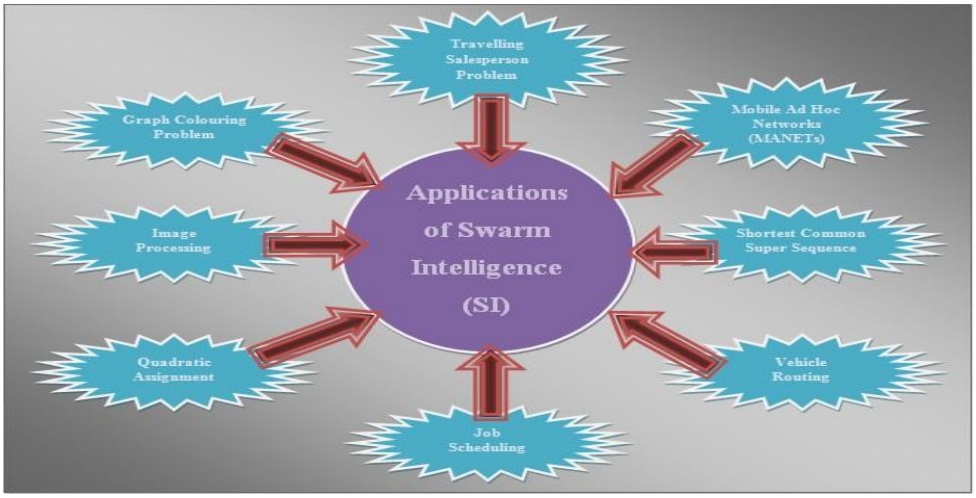


Figure 1.7. Applications of Swarm Intelligence (SI)

Let us consider an example of ant colony [13] to understand the working of SI. Goss et al. [14] perform this double bridge experiment as shown in Figure 1.8. In this a number of ants are placed at one end of bridge and some food source is kept at another end. Now initially, the ants pick up the random bridge to follow, but after some iterations, all the ants start to follow the same small length bridge.

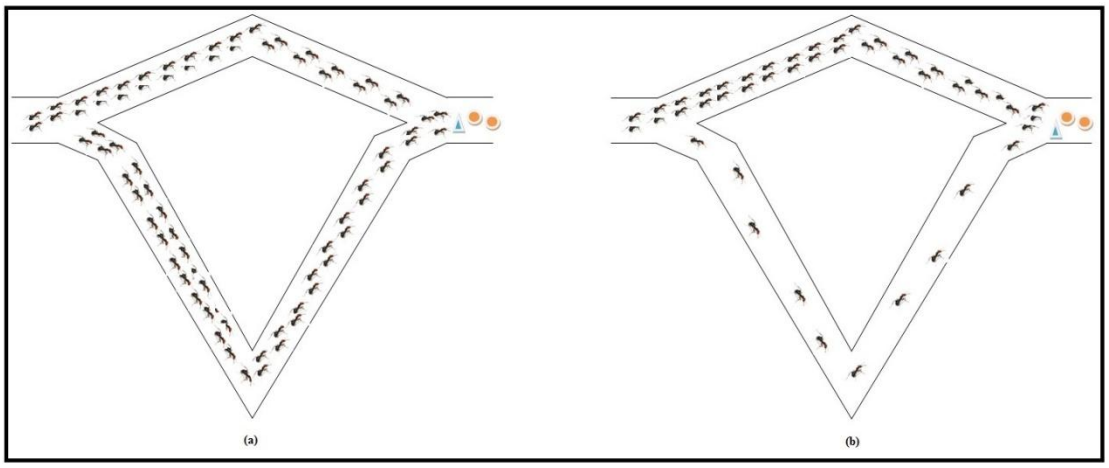


Figure 1.8. Double Bridge Experiment
(a) Initial State of the Experiment
(b) State of the Experiment After Some Time

This is how it happened. Initially all the ants start to follow the bridge randomly, but after some time based on the distance of the ants from the food source, they leave a pheromone value on the particular bridge they followed. Now after first iteration, they communicate with each other about the quality of the path they followed, and before following the path again or picking the bridge to follow, they evaluate the quality of the bridge based on the pheromone value. In this way they all started following the same bridge.

1.7. Firefly Algorithm

Firefly algorithms is swarm intelligence based nature inspired meta-heuristic algorithms [15]. Here meta-heuristic is bigger level solution for optimization of a problem by selecting a lower level heuristic. Meta-heuristic is used where information is partially available. Firefly algorithm had been proposed in 2008 by Xin-She Yang [16] and has very vast applications because of its efficiency in optimized solutions.

1.7.1. Overview of Firefly Algorithm

Firefly algorithm, is a nature inspired meta-heuristic algorithm, which was developed considering the flashing patterns of fireflies and their behaviour. Most of these fireflies produce recurring and short flashes. For particular species of firefly, the prototype of rhythmic flashes is exclusive. The assumption made in firefly algorithm is that all the fireflies are of same sex. Two primary functionalities are implemented by these flashing patterns. One is to attract other firefly, i.e. one firefly is interested in communicating to other firefly, and second is that these flashes can also be used as to warn other firefly of the harshness of the firefly.

The effectiveness of the algorithm comes from the communication of these fireflies, i.e. how effectively they communicate with each other. Their communication determines how effectively they provide optimized solution to the problem. The communication between a pair of fireflies depends upon their recurring flashes, the rate of flashes and intensity of the flashes. The fireflies can also make the system behave self-organized by synchronizing their flashes with each other. This makes the system optimized and intelligent.

The effectiveness of the algorithm depends upon the communication among the fireflies, which, in turn, depends on the flashing pattern that is directly proportional to intensity of the flashes. As we know, the intensity is inversely proportional to the distance, i.e. $I \propto 1/r^2$, more the distance between the fireflies less will be the intensity. Because of these stats, fireflies are only visible to a limited region. The intensity of the flashes is adjusted in such a way that the objective function gives optimum value in both the cases of maximization and minimization.

1.7.2. Basic Firefly Algorithm

The following basic rules are used to define basic firefly algorithm (Figure 1.9).

- All the fireflies are of same gender, so the attractiveness between the fireflies depends only on the intensity of the flashes regardless of the sex of the firefly.
- Attractiveness between the pair of firefly directly depends on the brightness of the firefly, i.e. which firefly will move to which firefly depend on their respective

brightness. The less brighter firefly will move closer to the more brighter one and if the intensity of both the fireflies are equal, then they will move independently.

- The brightness of the firefly can be evaluated from the objective function to be maximized or minimized.

Based on the rules, the firefly algorithm can be stated as pseudo code presented in figure 1.9.

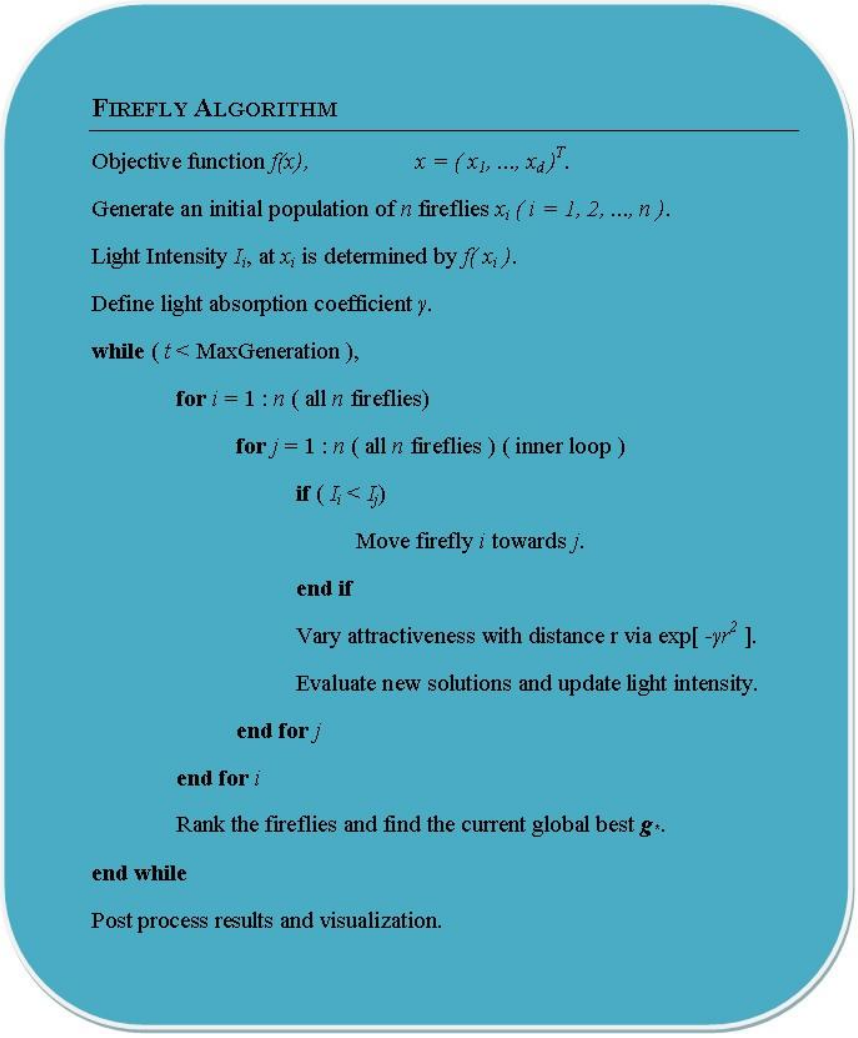


Figure 1.9. Firefly Algorithm

1.8. Organization of the Thesis

- The first chapter elaborate the details of the MANETs and the basic firefly algorithm. It include the classification of routing protocols in MANETs and the issues in MANETs. The introduction of firefly algorithm gave the background and the formulation of the firefly based systems. It helps in understanding the

behaviour of fireflies, i.e., process of indirect communication between fireflies through the intensity of the flashing patterns.

- The second chapter summarizes the various swarm intelligence based algorithms for MANETs. The rigorous literature study of many existing swarm intelligence based MANETs algorithms has been done.

- In the third chapter, the problem statement for the thesis is described and various objectives have been formulated for the work.

- The detail of the proposed algorithm is mentioned in the fourth chapter. A detailed example is also explained with static basic requirements.

- Fifth chapter explains the implementation environment and the simulations performed. This chapter also evaluates the performance of proposed algorithm with respect to various parameters by presenting the comparative results of the proposed algorithm. The proposed algorithm is compared with other existing algorithm AOMDV in terms of various performance metrics like throughput, packet delivery ratio, packet loss, and end to end delay.

- The sixth chapter concludes with the findings of the proposed algorithm. The results obtained from the previous chapter helped in concluding that the proposed algorithm perform better than the other algorithm used for comparison. This verifies the proposed algorithm. Lastly, the directions for the future enhancement of the proposed algorithm has also been stated.

Golshali *et al.* (2008) proposed an Ant Colony Optimization (ACO, class of SI) based routing algorithm known as Node-Neighbour-Number-Algorithm (NNNA) [17]. The algorithm possesses both proactive behaviour (path installation) and reactive behaviour (path maintenance). In this algorithm a list is created at each node which contains it neighbour and a special number which is given to every node in the list. The next hop is selected based on the intensity that is the number of neighbouring nodes. The main goal of the reported algorithm is low average end-to-end delay and high packet delivery ratio in dense and spread out network with high speed moving, the algorithm does so by reducing the routing overhead which is caused by almost 50% lesser broadcast.

Ko and Vaidya (2000) proposed Location-Aided Routing (LAR) [18] protocol for MANETs. The protocol uses the geographical information of the node to faster the route discovery. The location information helps to limit the search area (known as request zone) for the new route in ad hoc network and thus reducing the number of hello messages. Based on location information provided by GPS (Global Positioning System), two kind of zones expected and request zone are used. In expected zone, the source node predicts the location of another node at time t1 provided the source node knows its location at time t0. In request zone, its sends request to know its location information. The protocol mainly focuses on reducing the routing overhead.

Camara and Loureiro (2000) proposed an ACO based GPS (Global Positioning System) routing algorithm known as GPS/Ant-Like Routing Algorithm (GPSAL) [19]. Ants collects the location information of the node and disseminate in the network. In this algorithm the ants follow different routes and the dissemination of location of a node directly depends on the number of different routes explored. The mobile host are of two types fixed and mobile. The algorithm works on two basic rules, first is, the ant leaving a node carries the most updated information along with it, and second is, the location of destination node is assumed as the node which had not been updated for the longest period of time. Fixed hosts are used wherever possible to route the packets. The main goal of the protocol is decreased overhead of

the network and main cause for this is using the location information of the mobile host.

Marwaha *et al.* (2002) proposed a hybrid routing protocol known as Ant-AODV [20]. The protocol applies Ant technique on AODV (Ad Hoc On-Demand Distance Vector Routing Protocol). The other Ant-based routing algorithm had some flaws like the lesser number of ants are available in the network for route discovery, and only the ants knows the routes to the destination and the source had to wait for ant to send the packet. In this algorithm, the nodes are also given the ability to begin route discovery as needed and this results in higher node connectivity which gives lower route discovery latency. It reduces the time for route discovery and end-to-end delay, as the result decreases the number of route discovery.

Caro and Dorigo (1998) proposed a mobile-agents-based routing algorithm called AntNet [21]. The algorithm is Ant based and mainly uses the shortest path criteria. In this algorithm, every ant finds it own path to the destination and during this creation of path, the information attained by forward ant is communicated with backward ant i.e. the ant in the opposite direction and this information is used to update the routing information. Two parameters are basically updated by this algorithm, first one is the probability value of choosing a node as the next node (the chosen node is made the increased probability and probability for other nodes is decreased) and the second one is the trip time to every other node from the node to be updated. This algorithm basically works on two parameters, the size of the path to the destination from the source and the traffic density on the path. The backward ant updates the routing table depending on the round trip time of the forward ant.

Baras and Mehta (2003) proposed an Ant based Probabilistic Emergent Routing Algorithm (PERA) [22]. This algorithm uses the flooding approach for finding the path from source to destination. Forward and backward agents are used for route discovery. The probability distribution for all neighbours is adjusted at each node. The neighbour sends the request ahead and at some time delivers the packet. The pheromone values are maintained for each particular destination at each node. The concept of sequence number, node_time and ant_time is used for routing. The request with maximum sequence number and the time to reach the destination i.e. node_time-ant_time pair is used to proceed further. Single best route from the multiple available

routes is selected. The algorithm when compared to AODV reduces the end-to-end delay.

Yuan and Xiang (2004) proposed an ACO based hybrid routing algorithm known as Adaptive improvement (ARAAI) [23]. This algorithm mainly works on mobile and dynamic nature of MANETs. The algorithm has the key features such as self configured, distributed and self built. The pheromone deposited by the ants on the paths is used to select the neighbour as the next hop, more the pheromone, more is the chances of the path to get selected. The pheromone value is updated for a particular pair of source and destination, where source being the current node whose table to be updated. The algorithm works as reactive on start of session and the path is discovered. The ant here used are known as reactive ants, and it acts as proactive when session is progressing and paths are maintained and enhanced using proactive ants.

Sivakumar and Bhuvaneswaran (2007) proposed Multi agent Ants based Routing Algorithm (MARA) [24]. This routing algorithm is based on ACO and is a hybrid classified algorithm that requires lesser number of route discovery because of its multicast nature and its multi agent concept. This algorithm reactively setup the path and proactively maintains the same. In comparison to AODV, it outperforms in terms of delivery ratio, average delay and delay jitter.

Wang *et al.* (2009) proposed a Hybrid Ant colony optimization routing algorithm for mobile ad hoc networks (HOPNET) [25]. The ACO based routing protocol works on key concept known as zone routing framework. The whole network is divided among zones where each node belongs to individual zone. The nodes within the zone interact proactively while the nodes in different zones interact reactively. The proposed algorithm improves average end-to-end delay and delivery ratio when compared to AODV. The algorithm also outperforms AntHocNet when the zone radius is made greater than before.

Garg *et al.* proposed an Ant Colony Optimization routing (AOCR) protocol based on ACO for multi-hop networks. The authors provide a probabilistic technique in the form of AOCR for solving computational problems which are reduced to find good paths by graphs. Route maintenance is done periodically in this algorithm through

data packets to retain optimal path. This routing protocol is well adaptive, efficient and scalable. The main focus of this protocol is to reduce overhead for routing .

Osagie *et al.* (2008) proposed an ACO based imProved Ant Colony Optimization routing algorithm for mobile ad hoc NETworks (PACONET) [26]. Forward Ants (FANT) are used to find the routes in the network and Backward Ants (BANT) are used to update the same.

Karaboga has shown a comprehensive review of Artificial Bee Colony to provide a comprehensive survey of research on ABC [27]. Karaboga proposed a population-based evolutionary and stochastic method named Artificial Bee Colony (ABC) algorithm in the year 2005. ABC algorithm is simple and very flexible as compared to other swarm based algorithms. Because of its good convergence properties, this method has become very popular and is widely used. A system of comparisons and descriptions is used to designate the importance of ABC algorithm, its enhancement, hybrid approaches and applications.

Younes *et al.* proposed an improvement of the ACO meta-heuristic by using the method Artificial Bees Colony (ABC) [28]. In this an ABC algorithm is united with ACO, to solve the Economic Power Dispatch (EPD) problem. In this algorithm Bee colony is used for finding best values and these best values are used to improve the ACO. A hybrid method for ABC-ACO suggested in this paper automatically and dynamically adjust the values of the pheromone factor weight and the weight of heuristic factor that have a strong influence on the articulation intensification diversification of research in ACO. The main purpose of this algorithm is to free the user of feeling difficulty in setting parameters in ACO along with improving performance of overall cost function. The feasibility of the proposed approach was also tested on IEEE 57-bus system and it gave better results.

Hoolimath *et al.* proposed a routing algorithm called Optimized-Termite (Opt-Termite) [29]. This algorithm is bio-inspired routing algorithm for MANETs which behaves like real termites. Opt-Termite decreases the control packet overhead by using stigmergy for self-organization. The main concern of this algorithm is balancing the load. In this technique a reduced amount of loaded nodes are selected for traffic to

reach the final destination. The routing information is dependent on the movement of packets at each node and the routing table is updated accordingly .

Suguna *et al.* compared Bee-Ant Colony Optimized Routing (BACOR) with scalable Mobile Ad Hoc Networks (MANETs) routing protocol based on pause time. In this, the authors proposed an approach based on swarm intelligence for on demand routing protocols. This algorithm takes the advantages of both Ant Colony and Bee Colony Optimization for foraging of bees to calculate the fitness function and ability of ants to solve complex problems and named this algorithm as Bee-Ant Optimized Routing (BAOCR). Based on the estimated delay, signal strength and residual energy of the nearby nodes, BACOR finds the efficient node and sends the data packets through that node. The results of BACOR are compared with the other mobile routing tactics and this algorithm provides better results in terms of quality parameters as delay signal strength and energy [30].

Sharvani *et al.* used ant colony optimization for efficient stagnation avoidance for MANETs with local repair strategy [31]. In this, they developed an efficient routing algorithm called "Modified Termite algorithms" (MTA) for MANETs. MTA is developed by adopting efficient pheromone evaporation technique addresses to load balancing problems. By including QoS, efficient route maintenance, 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. The stagnation problems overcame by fine tuning of the Pheromone concentration based on node stability factor. The MTA implemented on MANETs with fine tuning of pheromone concentration shows significant increase in the throughput ~76% with increase of faulty nodes as compared to termite algorithms.

Gunes *et al.* [2002] proposed Ant-Colony-Based Routing Algorithm (ARA) [32] for multi-hop ad-hoc networks. This on-demand routing algorithm is ant colony based meta heuristics. In this the ant communicate with each other indirectly. Two kind of ants are used FANT and BANT, where FANT keep track of path from source, and BANT from destination. Sequence numbers are used to differentiate the packets originated from a common source. The algorithm mainly decreases the routing overhead.

Caro *et al.* [2005] proposed a hybrid routing algorithm known as AntHocNet [33]. This routing algorithm mainly focuses on self-organising behaviour of ant colony and the shortest path acquisition. The algorithm reactively setup the path using reactive forward and backward ants and the paths are maintained proactively using proactive backward ants. When compared with AODV, this algorithm proves itself better in respect of packet delivery ratio, average end-to-end delay and jitter.

Gudakahriz *et al.* proposed a reactive routing protocol named as Nature Inspired Scalable Routing (NISR) Protocol for MANETs [34]. The NISR is based on the combination of bee and ant colony optimization and TORA. The algorithm first searches for all the available paths to the destination from the source and the shortlist only one path with minimum hop count i.e. the shortest path available. And the pheromone value is increased by 1 only for the nodes on the path selected.

Chapter 3

Problem Statement and Objectives

3.1. Problem Statement

The mobile ad hoc networks are the infrastructure less networks constructed without any fixed infrastructure such as base station, tower, redirection switches and routers. Mobile ad hoc networks are the temporary wireless networks. All the routing information is managed by the node itself. The main task of MANETs nodes is to relay the data and information from source to destination. The performance of MANETs is calculated in terms of efficient transmission of data between the nodes without any loss.

Many routing protocols are there to improve the efficiency such as reactive and proactive protocols but still there is need of improvement. The performance results of routing protocol AOMDV are very weak in terms of quality of service.

The research work in this thesis is focused on the formulation of an firefly based algorithm by combining reactive and proactive nature of MANETs routing algorithm, because both existing proactive and reactive routing protocols prove to be inefficient under these unknown circumstances of MANETs. To do this, a new approach is suggested using firefly algorithm based on swarm intelligence. This approach calculates the best path for MANETs. This approach is combined with the AOMDV protocol to enhance its performance. Better throughput, packet delivery ratio and minimum packet loss were obtained during analysis.

3.2. Objectives

This thesis presents the novel, dynamic and adaptive routing algorithm for MANETs inspired by the behaviour exhibited by the fireflies. Various variants of firefly algorithm have been thoroughly studied before implementing the proposed work. After formulating the problem, following objectives were identified.

- To study and review various firefly optimization techniques for different problems.

- To design and develop efficient firefly based routing algorithm for MANETs.
- To verify the proposed firefly based algorithm and compare it with standard MANETs routing protocol.

Chapter 4
Proposed Work

4.1. Firefly Algorithm based Optimized Routing over MANETs

There are many research proposals in the past where Swarm Intelligence (SI) has proved its positive effect on routing in MANETs. The Firefly algorithm (A SI approach), where fireflies use the flashes to communicate with each other. The higher the intensity and frequency of the flashes between the fireflies (here denotes the Forward firefly FFLY and Backward firefly BFLY), higher the probability of selection of path between the nodes. The intensity and frequency of the firefly depends directly on the objective function. The higher the value of objective function, higher will be the intensity and higher will be the probability of choosing the node. So, among all the neighbouring nodes, the node with highest intensity will be given highest preference over others, and a sorting list will be created based on the intensity of the node.

The flowchart for the routing algorithm is shown below in Figure 4.1. When the route acquisition request is demanded, the algorithm will start searching for the optimum path from source to destination node. A sorted list is created for all the nodes in the network. Initially, the sorted list is sorted based on the objective function value. The objective function value for each node is calculated according to the initial value of the nodes. The Forward firefly (FFLY) is created at the source node. Then this FFLY is forwarded to the neighbouring node with higher objective function value. Then the selected intermediate node is checked whether it is a destination node or not. If no, then FFLY is forwarded to its neighbours and next neighbouring intermediate node is selected, and if yes, i.e. the intermediate node is the destination node, then the FFLY is destroyed and the Backward Firefly (BFLY) is created at the destination node. The BFLY then will follow the reverse path stored in FFLY (using STACK). On the reverse path, for every node till the source, the BFLY will calculate the firefly factor and update in the node value, so that next time the firefly factor plays its role in calculating objective function.

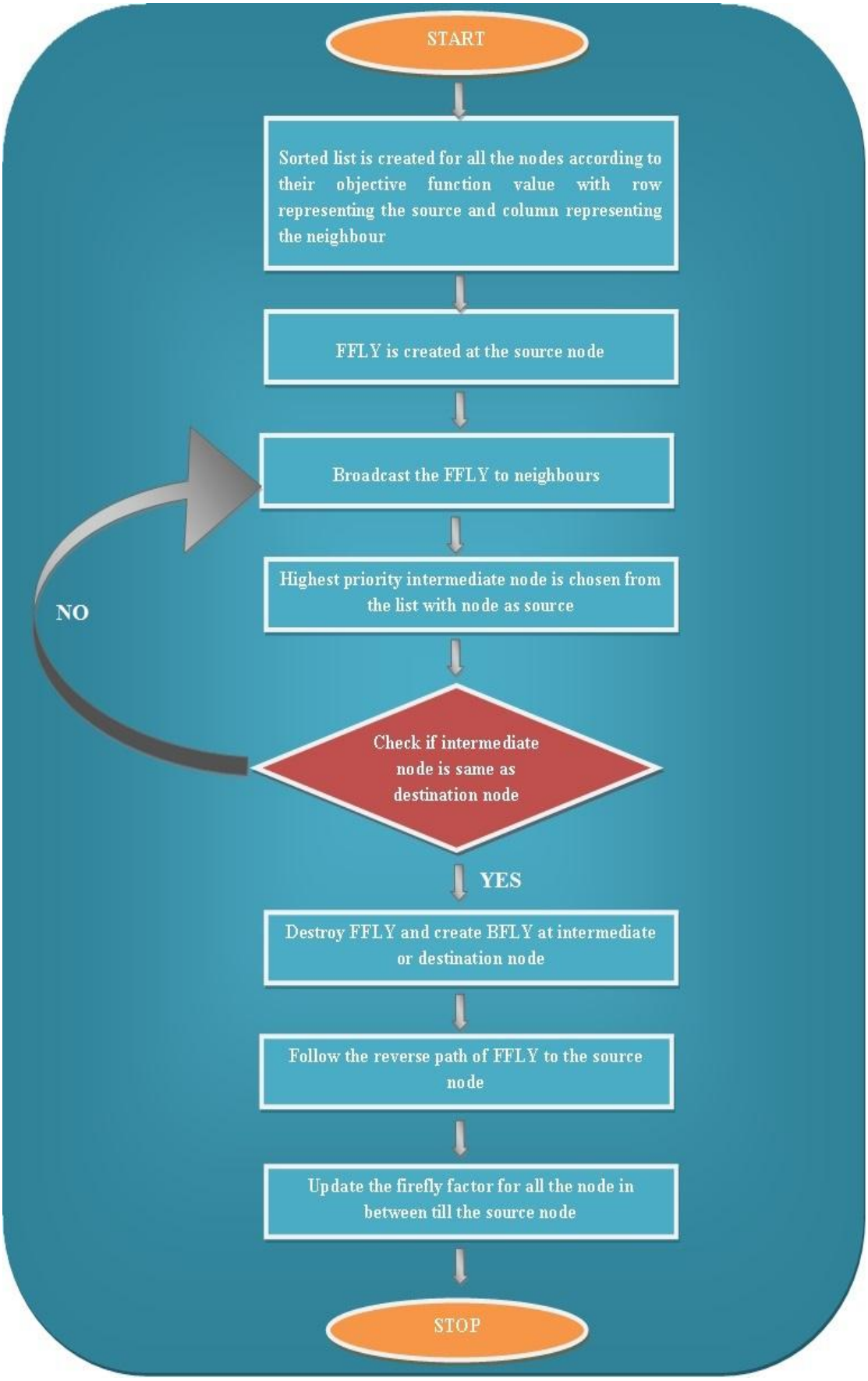


Figure 4.1. Flowchart of Proposed Algorithm

4.1.1. Data Structure Used

Two sets of homogeneous mobile agents that act as artificial fireflies called, FFLY and BFLY are used. But they are separately situated in the environment in different fashions such that is, they can sense different inputs and can produce different, independent outputs. They can be broadly classified as deliberative agents, because they behave reactively retrieving a pre-compiled set of behaviours, and also maintain a complete internal state description. The proposed algorithm uses set of artificial fireflies and replaces the typical routing table in traditional algorithms with the table of probabilities called the flash table. The following data structure is used by these fireflies which is maintained for each node.

- Destination
- Sequence Number
- route list { {next-hop-1, link-quality-1, link-availability-1, firefly-factor-1}, {next-hop-2, link-quality-2, link-availability-2, firefly-factor-2}, ... {next-hop-n, link-quality-n, link-availability-n, firefly-factor-n} }
- 2-D Swarm Array
- 1-D Local Best Array

4.1.2. Route Acquisition

When the route acquisition request is demanded, the algorithm shown in Fig 4.2 will start searching for the optimum path from source to destination node. A sorted list is created for all the nodes in the network. The sorted list is created on the objective function value of the node, where the row represents the source node and the column represents its neighbour.

The respective value is calculated based on the link availability, quality and the firefly factor of the link between them. Initially, the firefly factor for all the nodes is kept one. We apply firefly algorithm for every neighbouring node and use the global maximum value. For every intermediate node, the swarm array is initialized with 50 rows and 5 columns. These 5 columns are the input parameters for calculating link availability and link quality.

ROUTE ACQUISITION ALGORITHM

1. Begin route discovery at source node.
2. Source node will calculate objective function for all its neighbours using the formula

$$f(x_i) = f(x_i - 1) + \{(Link\ availability * AF1) + (Link\ quality * AF2) + (Firefly\ factor * AF3)\}$$

3. and create FFLY packet at source node and flood to the neighbour with maximum objective function value.
4. Neighbours receiving the FFLY packet for a destination, will check if route to destination is available or not.
5. If no, they will repeat from step 2 onwards by calculative objective function value for its neighbour.

Figure 4.2. Route Acquisition Algorithm

A lower bound and upper bound for every input parameter is initialized. Lower bound will be the lowest value that the input parameter can have and upper bound will be the maximum value that the input parameter can have. Then the swarm array is calculate using [35] as

$$swarm(i,j) = LB(j) + (rand(1)\%2) * (UB(j) - LB(j)) \quad (4.1)$$

Then for every row, another array swarm_func_value is calculated using the objective function. Then the maximum of the array is used as the objective function value.

The value for objective function to be maximized is calculated for every row according to the formula

$$f(x,y) = AF1 * link\_avail(x,y) + AF2 * link\_qual(x,y) + AF3 * firefly\_factor(x,y) \quad (4.2)$$

where, AF1, AF2 and AF3 are the adjusting factor and are normalized like,

$$AF1 = \frac{link\_avail(x,y)}{link\_avail(x,y) + link\_qual(x,y) + firefly\_factor(x,y)} \quad (4.3)$$

$$AF2 = \frac{link\_qual(x,y)}{link\_avail(x,y) + link\_qual(x,y) + firefly\_factor(x,y)} \quad (4.4)$$

$$AF3 = \frac{firefly\_factor(x,y)}{link\_avail(x,y) + link\_qual(x,y) + firefly\_factor(x,y)} \quad (4.5)$$

and link availability (link_avail(x,y)) [36] is calculated as

$$link\_avail(x,y) = p1 * p2 \quad (4.6)$$

where p1 is calculated for source and p2 for its neighbour using the formula

$$p1\ (or\ p2) = {}^{1}\!/_{2} * \left(1 - \left(\left(-2 * range^2 / \alpha_{eq}\right) * \left(e^{\left(-2*range^2/\alpha_{eq}\right)}\right)\right)\right) \quad (4.7)$$

where, range is the mobile node range,

α_{eq} is calculated as

$$\alpha_{eq} = 2 * t * \left(\frac{\sigma^2 + \mu^2}{\delta}\right) \quad (4.8)$$

where, t is time duration for epoch, σ is the variance speed, μ is the mean speed and 1/δ is the epoch time.

Link quality (link_qual (x,y)) is calculated as

$$link\_qual(x,y) = {}^{1}\!/_{P_{SR}} \quad (4.9)$$

where, P_{SR} is the probability of successfully sending and receiving the packet in a single attempt and is calculated as

$$P_{SR} = (1 - R_{LOSSFP}) * (1 - R_{LOSSRP}) \quad (4.10)$$

where, R_{LOSSFP} and R_{LOSSRp} are loss for forward and reverse packet and are calculated as

$$R_{LOSSFP} = {}^{(N - N_x)}/_{N} \quad (4.11)$$

$$R_{LOSSRP} = {}^{(N - N_y)}/_{N} \quad (4.12)$$

where, N_x, N_y are the hello packets received by x and y respectively and N is the number of packets sent by each node in fixed time.

4.1.3. Route Update

When the FFLY reached the destination node or that intermediate node which have a path to destination node, the algorithm shown in Figure 4.3 will destroy FFLY and create the BFLY at that node.

ROUTE UPDATE ALGORITHM

1. The BFLY is created at destination node, or at the node with the path to the destination node.
2. The BFLY will follow the reverse path of FFLY and update Firefly_Factor for all intermediate node till the source node using the formula

$$firefly\_factor(i) = firefly\_factor(i-1) + \beta_0 * e^{-y*r1*(firefly\_factor\ (i-1)+\alpha*\varepsilon)}$$

3. The source node Firefly_Factor will also get updated.

Figure 4.3. Route Update Algorithm

The BFLY then will follow the reverse path stored in FFLY. And on the reverse path, for every node till the source, the BFLY will calculate the firefly factor and update in the node value, so that next time the firefly factor will play its role in calculating objective function.

Firefly factor is updated [15] using the previous value as

$$firefly_{factor\ (i)} = firefly_{factor\ (i-1)} + \beta_0 * e^{-\gamma * r1 * (firefly\_factor\ (i-1) + a * e)} \quad (4.13)$$

where, β_0 is the attractiveness and is taken as 1, γ is the light absorption coefficient which is generalized to 0.5, r1 is the distance computed in terms of deviation between the nodes as

$$r1 = \sqrt{\frac{1}{k} \times \sum_{i=1}^{k} \left(UB(i) - LB(i)\right)^2} \quad (4.14)$$

where, k is the number of input parameters, a is the randomization parameter and e is the vector of random number drawn from a Gaussian distribution and both are equal to

$$a = e = rand()\%2 \quad (4.15)$$

4.2. Working of Algorithm with Static Network

Let us understand the working of the proposed routing algorithm via a static network of 10 nodes as shown in Figure 4.4. All these nodes are mobile nodes which have their own properties like range, movement speed, epoch time, etc.

Now suppose, a route is needed from source node 1 to destination node 10, then first of all, the network will be initialized with the nodes given the input parameter values as required by node data structure. Based on the link between the nodes, an adjacency list will be created. This list is a matrix formation of the network. Now, as the adjacency list of the nodes had been created and also the nodes are initialized, a sorted adjacency list from the adjacency list will be created based on the objective function value of each node which depends on the lower and upper bound of input

parameters. The cause for creating the sorted adjacency list is to have the neighbouring nodes sorted in preference order.

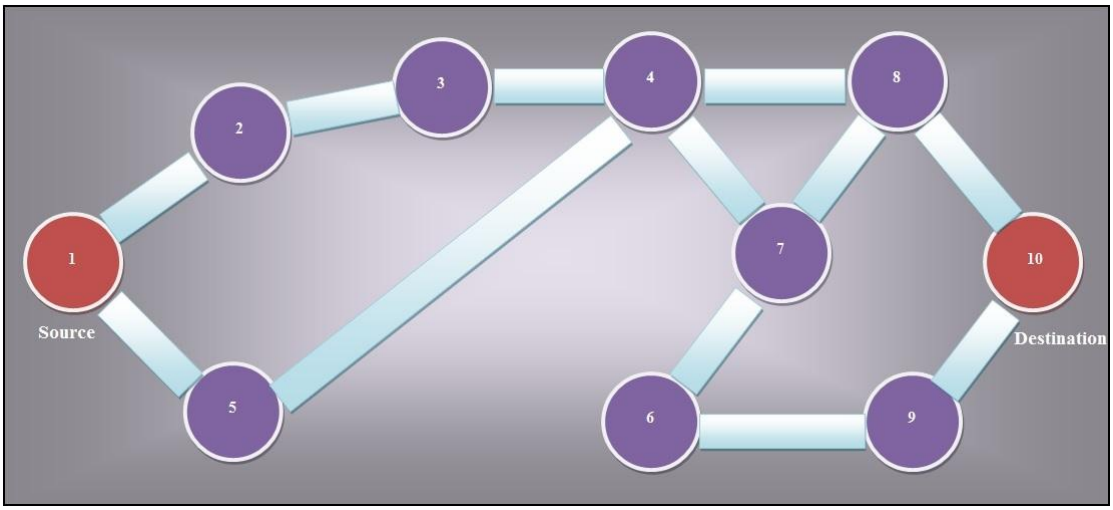


Figure 4.4. Static Mobile Ad Hoc Network

Now, the source node will look for its neighbours in sorted adjacency list. Suppose node 2 will evaluate to 12.5 and node 5 to 11.25, then node 2 will be selected and a FFLY will created at source node and forwarded to node 2, then it is checked whether node 2 is destination or not. If not, then FFLY will be forwarded to its neighbours i.e. node 3 only if this has not been visited already, so node 3 will be chosen and similarly the algorithm will go on repeating the above steps searching the path till node 10 i.e. destination is reached or the path to the destination is not found. Now suppose in future, if say node 2 does not deduce a path to the destination, then node 2 will be discarded and algorithm will backtrack and go on with node 5. Whenever the destination node is reached, the FFLY is destroyed and a BFLY will be created at the destination node (10) and it will follow the reverse path of FFLY. The BFLY will then update the firefly factor for the nodes on the reverse path. Hence next time, the selected route will be more effective.

Chapter 5

Performance Evaluation and Results

5.1. Working with NS2

5.1.1. Introduction to Network Simulator

Network simulator is the standard simulator used for designing of new protocols as per considered as best one to give the correct results. NS-2 is the simulator that works in three phases. One is the TCL scripting and other two are the high level language based such as C++ or JAVA [37]. Ns began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns has always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.

5.1.2. Beginning with NS-2

The first step is to install the NS-2. I have installed NS-2 on RED HAT LINUX. The presence of gcc compiler is must for working of the NS-2. The reason for this is that cross compiler is required to interpret the coding performed in NS-2; this compiler is provided through the gcc environment. To check the presence of gcc use the following command:

```
cd> gcc –v
```

Now if the message showing the configuration of gcc is displayed, this means that the gcc is correctly installed on the machine, in case there is error message then there is requirement of installing gcc again in correct format. After this installation, the second step is to install NS-2. This can be done by using simple Linux commands and then exploring the .install file present in the NS-2 folder. Then the installation will start. One can then validate or use it without validation. The command for validation is:

```
cd> ./validate
```

After validation, the next step is to check for ns if installed correctly. To do this, go to ns-2.3x folder and then type:

```
cd/ns2.3x>./ns
```

If, % sign appears then the NS-2 is installed correctly else you need to reinstall it. Here x denotes the version of the NS-2 used.

For working on MANETs in ns2 we have to create traffic files and scenario files for data traffic and for defining the position of mobile nodes

5.1.3. Scenario Files

The concept of setting up of the scenario for the animator in NS-2 is one of the key concern for getting the automated setting of the nodal structure thus, allowing the researcher to make the scenario that can handle the multi-functioning of the node. The scenario configured by the user is required to be placed as a tcl script file in the SCENE folder present under the MOBILITY folder present in the TCL folder. The scenario available in the NS-2 can be set for both wired and wireless mediums. The concept of MANETs is totally wireless, thus the explanation given below is regarding the configuration of the wireless medium. The scenario available is in two different versions- version 1 and version 2. The difference between both these versions is because of the number of parameters one can the user is required to be placed as a tcl script file in the SCENE folder present under the MOBILITY folder present in the TCL folder. The scenario available in the NS-2 can be set for both wired and wireless mediums. To create a scenario file we have to write the following command

```
setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed]
[-t simtime] \[-x maxx] [-y maxy] > [outdir/movementfile]
```

For example in this proposed algorithm 10,15,20,25, 30,50 nodes with p pause time 2, speed 30m/s, simulation time 1000 and area 1500x1500 are taken.

```
./setdest -n 10 -p 2.0 -s 30.0 -t 1000 -x 1500 -y 1500 > scen1500x1500-10-0-1000
```

There is no need to run the output. It is easy to redirect the output by opening file scen1500x1500-10-0-1000. The file gives the complete description about total

number nodes, with their initial positions and their movement form initial position to final position as shown below.

```
$ns_ at 2.000000000000 "$node_(0) setdest
90.441179033457 44.896095544010
1.373556960010"
```

This line depicts that at 2.0 second the 0th node starts to move at destination (90.44, 44.89) with the speed of 1.37m/s. We can explicitly change the direction and movement by changing these commands.

Directives for GOD are present as well in node-movement file. The General Operations Director (GOD) object is used to store global information about the state of the environment, network, or nodes that an omniscient observer would have, but that should not be made known to any participant in the simulation.

Currently, the god object is used only to store an array of the shortest number of hops required to reach from one node to another. The god object does not calculate this on the fly during simulation runs, since it can be quite time consuming. The information is loaded into the god object from the movement pattern file where lines of the form

```
$ns_ at 899.642 "$god_ set-dist 23 46 2"
```

are used to load the god object with the knowledge that the shortest path between node 23 and node 46 changed to 2 hops at time 899.642.

5.1.4. Traffic Files

The traffic generator is the file that is actually required to generate the traffic from the source towards the destination. The scenario set in the scenario file is incomplete without this traffic generator file as it decides the movement of data from the source towards the destination and this configuration also leads to movement of nodes at particular location following the mobility factor and thus moves the nodes to new set of location. This requires mobile transmission of data and this can be achieved through this tcl scripting. Again, this can be achieved through coding by an individual also, as described in the earlier portion of the paper but this can also be generated as a

single command in NS-2. The whole command is to be performed in the terminal if opened in LINUX or in the Cygwin terminal if being worked upon the window based NS-2. The terminal is to be opened at the location cmu-scen-gen as it was opened in scenario configuration. The sample code for configuration of the traffic generator file is as follows

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-
mc connections][-rate rate] > cbr-file
```

For example, a cbr connection files of 10 nodes, having minimum 5 connections, with a seed value of 1.0 and rate of 0.5 m/s. So command type is:

```
ns cbrgen.tcl -type cbr -nn 10 -seed 1 -mc 5 -rate 0.5 >
cbr-10-5-2-1
```

Where nn is the number of nodes, seed is the uniqueness of node, mc is minimum connection required and rate is the rate at which packets are sent from one node to others. The output of cbr-10-5-2-1 is shown bellow. It shows the connections of 2nd node to 3rd at time 82.5s with UDP connection setup between these nodes. Total UDP sources are 5 and total number of connections is 8 respectively

```
#
# 2 connecting to 3 at time 82.557023746220864
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
```

```
$ns_ at 82.557023746220864 "$cbr_(0) start"
```

In proposed algorithm, cbr type traffic file is created having 10, 15, 20, 25, 30, 35, 40, 45 and 50 nodes with mobility 10 to 30 Kmph and minimum connections 5 to 25 for each.

After setting of the scenario and the traffic generator for the MANETs animator, perform the following tcl scripting in the code file to include the scenario and the traffic generator file for working of the simulation.

```
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(x) 1500
set val(y) 1500
set val(ifqlen) 50
set val(seed) 1
set val(adhocRouting) AOMDV
set val(nn) 10
set val(cp) "../mobility/scene/ cbr-10-5-2-1"
set val(sc) "../mobility/scene/ scen-10-1500x1500-10-0-
1000"
set val(stop) 1000.0
```

- *Nam file :* Nam file is the visualization of node movements which shows how nodes interact with each other and how they send and receive packets. It has the following features.
  - Can be executed directly from a Tcl script
  - Controls include play, stop fast forward, rewind, pause, a display speed controller and a packet monitor facility.
  - Presents information such as throughput, number packets on each link, etc.

- Provides a drag and drop interface for creating topologies.

- *Trace File :* Trace file is a .tr file same as nam file that gives the additional information of messages that are sent or received using command prompt.

5.2. Simulation and Performance Evaluation

5.2.1. Experiment Setup

The performance of new routing algorithm has been compared with existing and known algorithm. The simulated traffic is Constant Bit Rate (cbr). Here, the results of the proposed algorithm is compared with the results of AOMDV. The proposed approach improves performance of quality metrics such as packet delay, packet delivery ratio, throughput and end-to-end delay. AOMDV perform better in terms of robustness when pause time network dynamics are down to 300s. As AOMDV use particular route during transmission with alteration in internet, the existing path will destroy that requires computation of new path which can cause network delay. But, in the proposed approach this thing does not cause a problem because of different routes. All results are very close to the simulation scenarios; this shows that the algorithm creates much less routing overhead for all considered mobility scenarios. From results it is proved that the proposed algorithm is better in all aspects like PDR, throughput and packet loss etc.

Performance of the proposed approach is analyzed and compared with AOMDV by using the network simulator NS-2.35. For simulation, different numbers of nodes are considered with an area of 1500 x 1500 sq.m. The other parameters configured for simulation are shown in the Table 5.1.

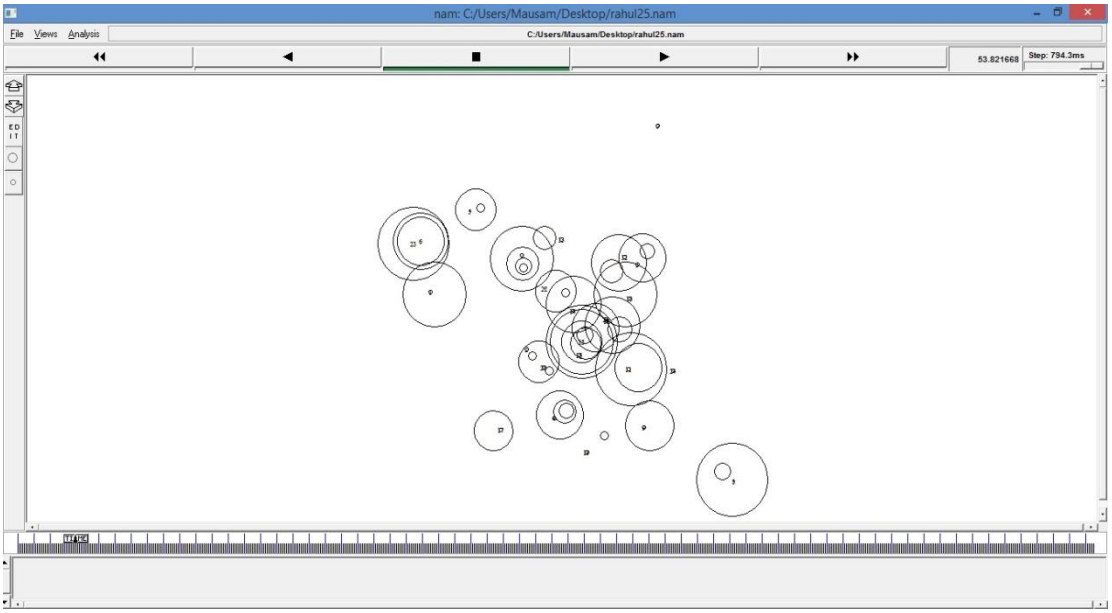
Table 5.1. Simulation Parameters

Parameters	Value
Version	2
Area	1500 x 1500 sq.m.
Nodes	10, 15, 20, 25, 30, 35, 40, 45, 50
No. of Fireflies	10, 15, 20, 25, 30, 35, 40, 45, 50
Maximum Speed	30 Kmph
Minimum Speed	10 Kmph

Simulation time	1000s
Simulation Runs	30
Minimum Connections	5-25
Connection Type	CBR
Pause Time	2 Sec
Total no of Packets	15000
Protocols Compared	AOMDV

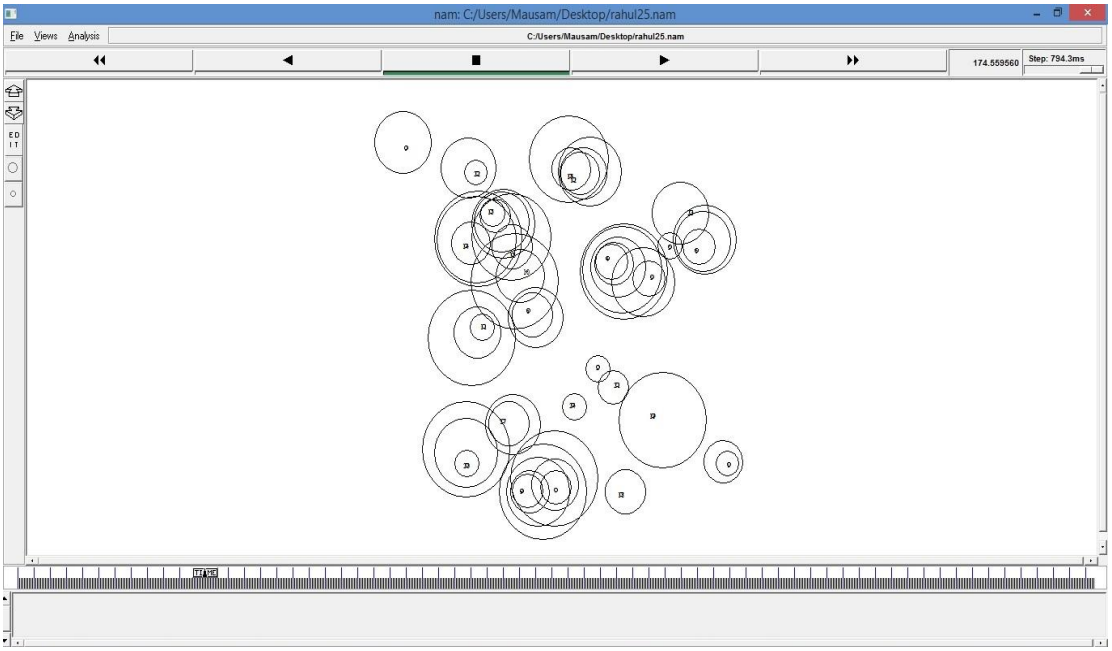
Below snapshots of nam file shows the movement of nodes. In this file, nodes broadcast requests to the other nodes to make a connection with them after which these nodes send their corresponding packets and data and get back the acknowledgement. The transfer of packets and acknowledgements at a particular moment are shown in snapshot 5.1, 5.2, 5.3 and 5.4. The nodes start sending packets after making connections through broadcasting.

Snapshot 5.1 shows the transfer of packets and acknowledgements at time t=53.821668. At this time, the nodes begin to create new path and as the time grows the nodes adaptively learn new efficient path.



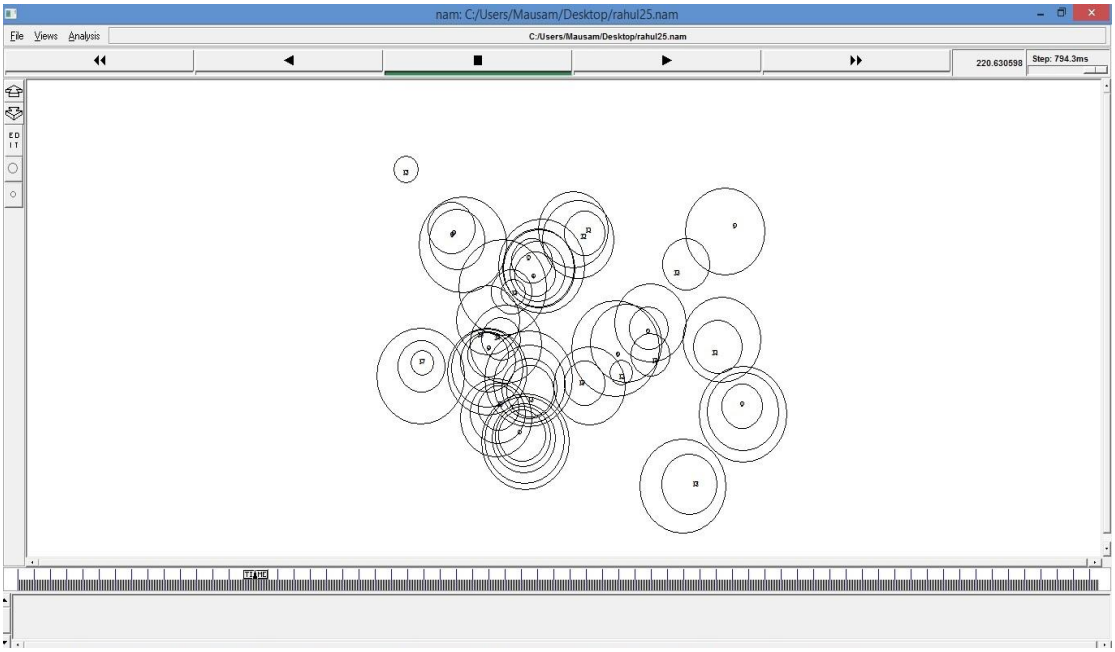
Snapshot 5.1. NAM file at time t = 53.821668s

Snapshot 5.2 shows the transfer of packets and acknowledgements at time t=174.559560. At this time, the nodes had learnt some new path, and thus more number of packets are transferred.



Snapshot 5.2. NAM file at time t = 174.559560s

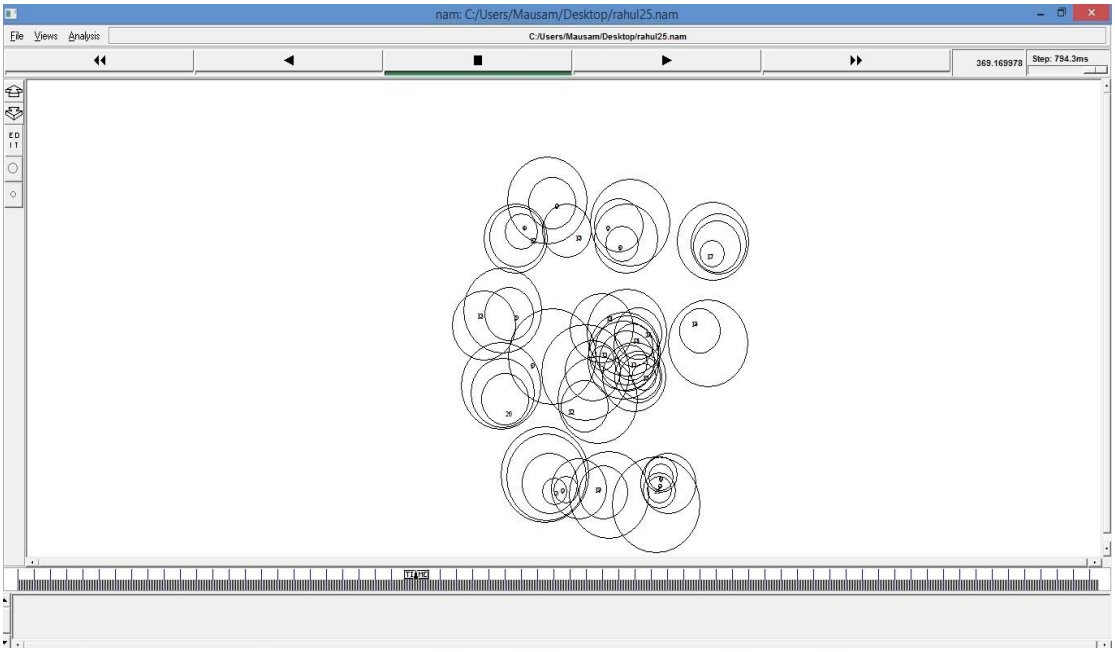
As firefly algorithm gives better optimized results with the time, the nodes keep learning efficient paths and more number of transmissions occurs. The transmission of packets at time t=220.630598 is shown is Snapshot 5.3.



Snapshot 5.3. NAM file at time t = 220.630598s

Snapshot 5.4 shows the transmission of packets at time t=369.169978. As shown in snapshot, more number of transmissions as compared to number of transmissions

shown previously. The reason for more number of transmission is that nodes learn more efficient paths with passage of time.



Snapshot 5.4. NAM file at time t = 369.169978s

A trace file of proposed algorithm operating over AOMDV protocol is shown below in snapshot 5.5 and snapshot 5.6.

ff_aomdv.tr (~/ns-allinone-2.34/ns-2.34/tcl/ex) - gedit

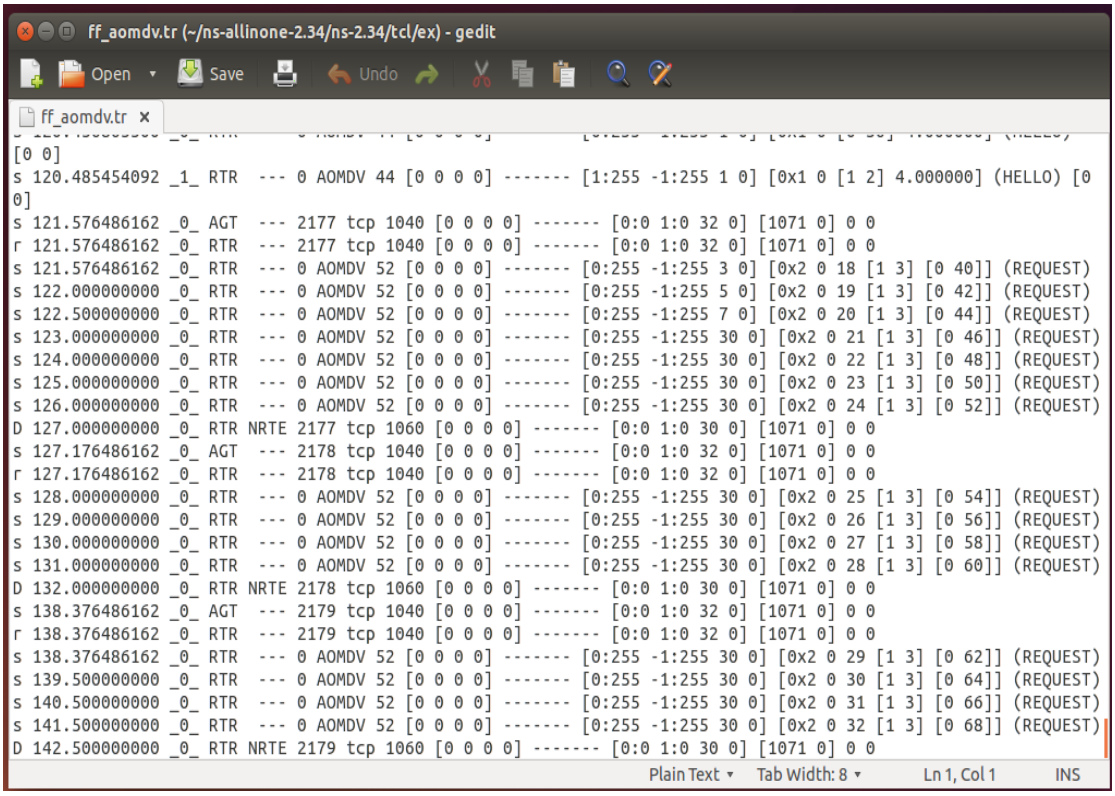
Open Save Undo

ff_aomdv.tr

```
M 10.00000 0 (5.00, 2.00, 0.00), (20.00, 18.00), 1.00
s 10.000000000 _0_ AGT  --- 0 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 10.000000000 _0_ RTR  --- 0 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 10.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 1 [1 0] [0 4]] (REQUEST)
s 11.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 2 [1 0] [0 6]] (REQUEST)
s 12.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 3 [1 0] [0 8]] (REQUEST)
s 13.000000000 _0_ AGT  --- 1 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 13.000000000 _0_ RTR  --- 1 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 13.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 4 [1 0] [0 10]] (REQUEST)
D 14.000000000 _0_ RTR NRTE 0 tcp 60 [0 0 0 0] ------- [0:0 1:0 30 0] [0 0] 0 0
D 14.000000000 _0_ RTR NRTE 1 tcp 60 [0 0 0 0] ------- [0:0 1:0 30 0] [0 0] 0 0
s 19.000000000 _0_ AGT  --- 2 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 19.000000000 _0_ RTR  --- 2 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 19.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 5 [1 0] [0 12]] (REQUEST)
s 20.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 6 [1 0] [0 14]] (REQUEST)
s 21.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 7 [1 0] [0 16]] (REQUEST)
s 22.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 8 [1 0] [0 18]] (REQUEST)
D 23.000000000 _0_ RTR NRTE 2 tcp 60 [0 0 0 0] ------- [0:0 1:0 30 0] [0 0] 0 0
s 31.000000000 _0_ AGT  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 31.000000000 _0_ RTR  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 31.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 9 [1 0] [0 20]] (REQUEST)
s 32.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 10 [1 0] [0 22]] (REQUEST)
s 33.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 11 [1 0] [0 24]] (REQUEST)
s 34.000000000 _0_ RTR  --- 0 AOMDV 52 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 0 12 [1 0] [0 26]] (REQUEST)
D 35.000000000 _0_ RTR NRTE 3 tcp 60 [0 0 0 0] ------- [0:0 1:0 30 0] [0 0] 0 0
M 50.00000 1 (390.00, 385.00, 0.00), (25.00, 20.00), 15.00
s 55.000000000 _0_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
```

Plain Text Tab Width: 8 Ln 1, Col 1 INS

Snapshot 5.5. Trace file_1



Snapshot 5.5. Trace file_2

5.2.2. Experimental Results

Parameters like packet delivery ratio, packet loss, throughput and end-to-end delay are evaluated to test the performance of proposed algorithm.

- Packet Delivery Ratio

Packet delivery ratio can be calculated by finding the percentage of successfully delivered packets to the total number of packets transmitted.

$$\text{Packet Delivery Ratio} = \frac{\text{number of packets received}}{\text{number of packets transmitted}} \times 100 \tag{5.1}$$

- *Effect on packet delivery ratio:* As shown in Figure 5.1, the results obtained from proposed algorithm is better than AOMDV algorithm. More number of packets are delivered by proposed algorithms. Packets delivery ratio is continuously increasing after a certain number of nodes.

Table 5.2. Packet Delivery Ratio from Different Experiments

Nodes	AOMDV	Proposed
10	32.11	48.8995
15	44.4444	52.5921
20	56.7788	66.2847
25	69.1132	69.9773
30	70.4476	73.2229
35	73.782	77.3625
40	76.1164	80.6081
45	78.4508	84.7477
50	80.7852	90.4403

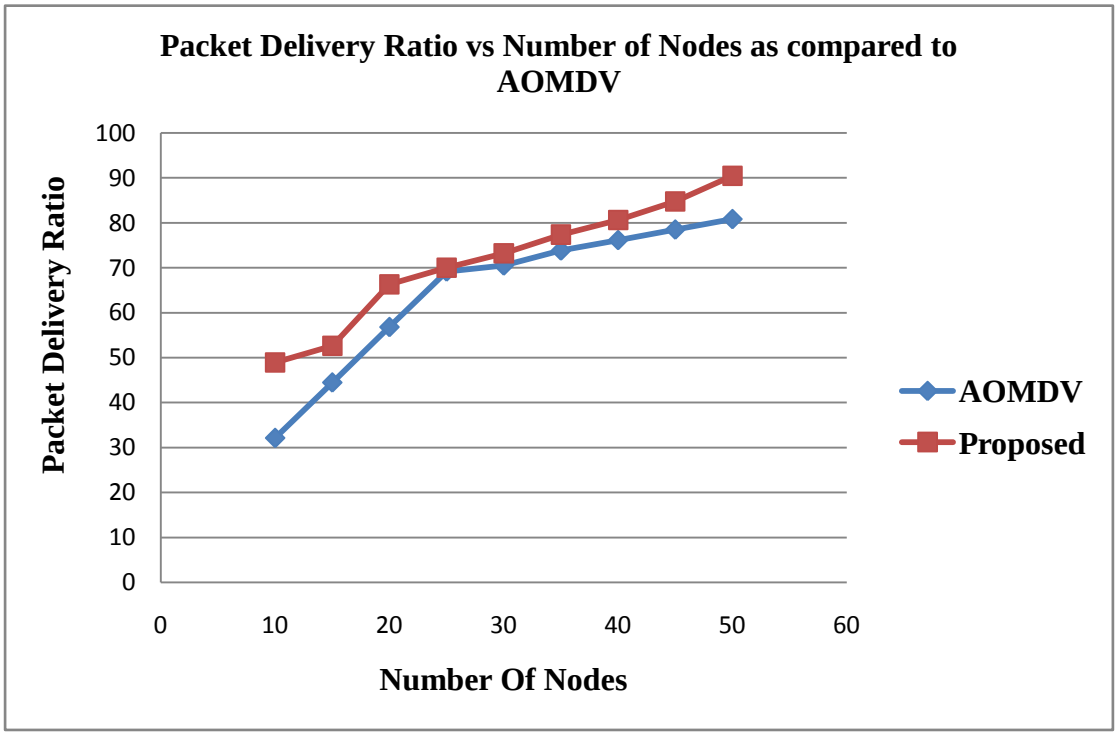


Figure 5.1. Packet Deliver Ratio on Comparison with AOMDV

- Packet Loss

Packet loss is the number of packets dropped while transmitting packets.

$$Packet\ Loss = \frac{number\ of\ unsuccessful\ transmissions}{total\ packets\ sent} \times 100 \quad (5.2)$$

- *Effect on packet loss:* The Figure 5.2 shows that the proposed algorithm loses lesser number of packets as compared to AOMDV algorithm.

Table 5.3. Packet Loss from Different Experiments

Nodes	AOMDV	Proposed
10	67.89	51.1005
15	55.5556	47.4079
20	43.2212	33.7153
25	30.8868	30.0227
30	29.5524	26.7771
35	26.218	22.6375
40	23.8836	19.3919
45	21.5492	15.2523
50	19.2148	9.5597

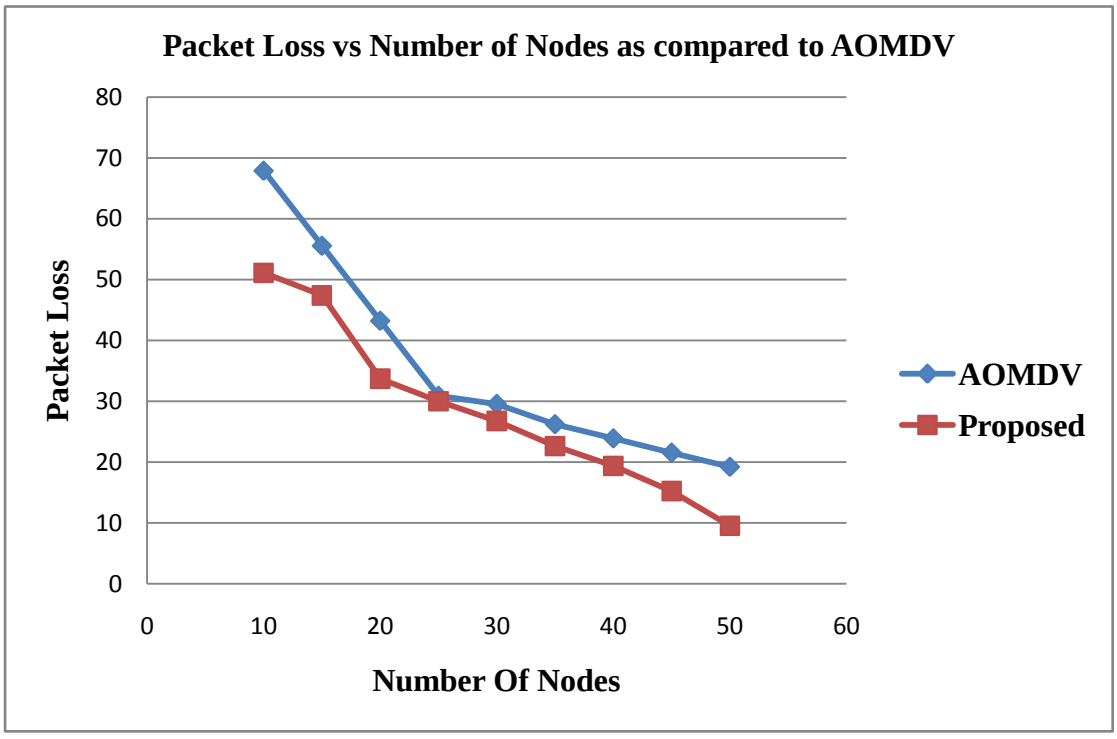


Figure 5.2. Packet Loss on Comparison with AOMDV

- Throughput

Throughput is amount of data (packets) delivered by the system per unit time and is measured in Mbps (Mega bits per second).

$$Throughput = {total\ number\ of\ packets}/{total\ time} \quad (5.3)$$

- *Effect on throughput:* More throughput is given by the proposed algorithm than AOMDV algorithm as shown in Figure 5.3 . The proposed algorithm gives more throughput as every time next hop is chosen on basis of several parameters using firefly algorithm. This helps in choosing best available neighbour node.

Table 5.4. Throughput from Different Experiments

Nodes	AOMDV	Proposed
10	1.1	2.247
15	1.7	2.547
20	1.9	3.247
25	2.36	3.347
30	3.1	3.847
35	3.43	4.147
40	3.8	4.447
45	3.9	4.547
50	4.2	4.747

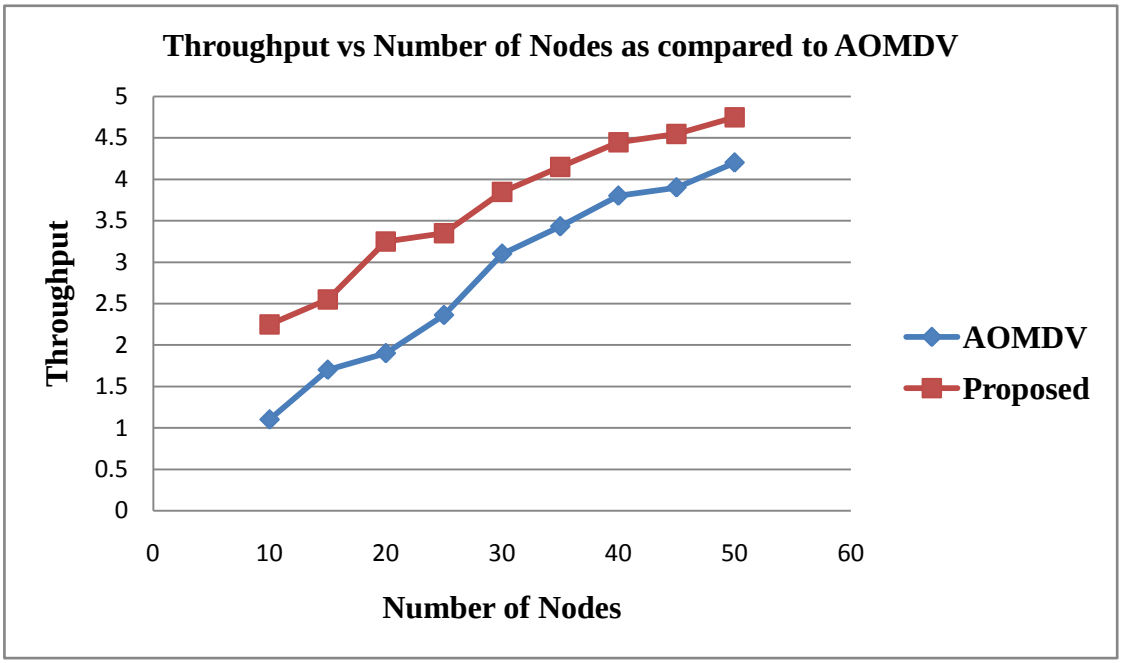


Figure 5.3. Throughput on Comparison with AOMDV

- End-to-End Delay (E2D)

E2D is the total time for the packet to reach the destination from source.

$$E2D\ delay = time\ from\ hop1\ to\ hop2 + time\ from\ hop2\ to\ hop3 + , \ldots , + time\ to\ destination \quad (5.4)$$

- *Effect on End-to-End delay:* The End-to-End delay for the proposed algorithm is lesser as compared to AOMDV algorithm as best among all the next hop is chosen at every step as shown in Figure 5.4.

Table 5.5. End-To-End Delay from Different Experiments

Nodes	AOMDV	Proposed
10	113.14	103.697
15	109.52	93.699
20	102.14	91.037
25	91.36	91.687
30	90.43	86.367
35	89.5	84.2904
40	87.57	81.2218
45	83.64	76.097
50	73.71	60.947

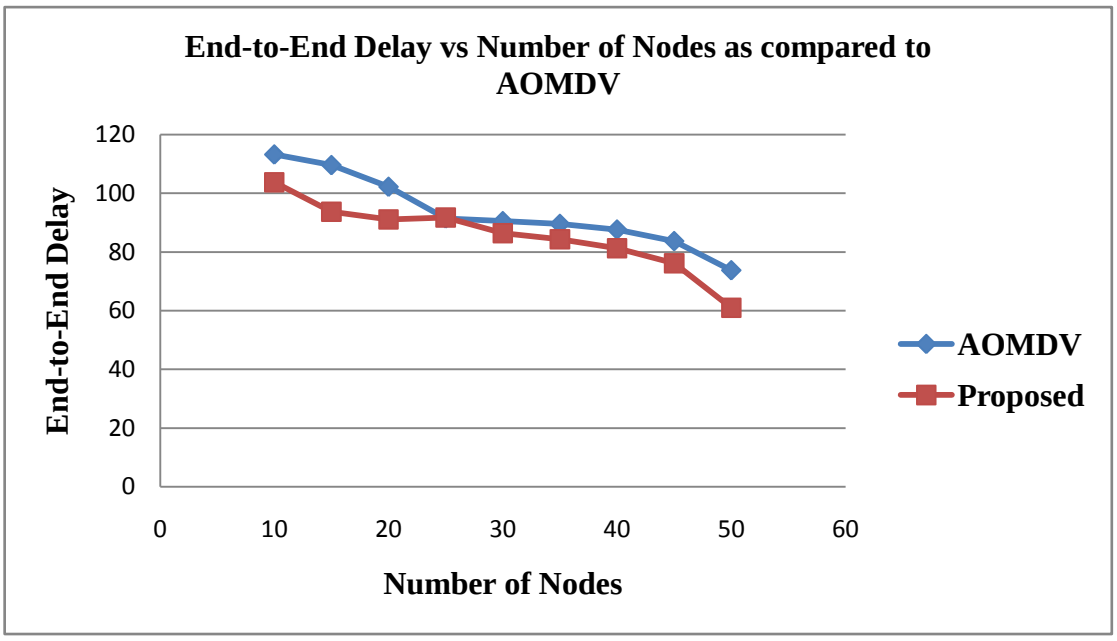


Figure 5.4. End-to-End Delay on Comparison with AOMDV

Chapter 5

Conclusion and Future Scope

Mobile Ad-Hoc Networks (MANETs) is the collection of wireless mobile nodes that make a momentary network not using any centralized access points or centralized administration. MANETs comprise of mobile nodes that have capabilities of communicating with each other using packet radios over a shared wireless medium. But due to small range of wireless network, it needs multiple network hops to exchange data and information with other nodes over the network. The main problem with such network is mobility of the nodes, the nodes are free to roam arbitrarily, thus the topology changes time to time, and route acquisition in such a network is always being the tedious task. For an efficient network formation, mobile nodes should be capable enough to utilize the available query resources.

We have proposed firefly based routing algorithm for optimized transmission over MANETs. It is a reactive routing algorithm which applies firefly between every pair of a node and its neighbour. Based on link availability, link quality and firefly factor, next hop is selected each time a path is to be established. This selection is made using existing firefly algorithm.

The results showed that the proposed algorithm was capable to handle network latency and offers better network lifetime, and it can promptly adapt to a changing network status.

The probability distribution changes after every iteration makes real time implementation of these algorithms a challenging task thus in future an efficient probability distribution model can be formulated that can optimally utilize the features of the proposed scheme in real time environment.

REFERENCES

1. Aarti, S. Tyagi. “Study Of Manet: Characteristics, Challenges, Application And Security Attacks”. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 5, pp. 252-257, 2013.

2. D. Sensarma, K Majumder. "An efficient ant based qos aware intelligent temporally ordered routing algorithm for MANETs". International Journal of Computer Networks & Communications (IJCNC), Vol. 5, No. 4, pp. 189-203, 2013.

3. D. Garg, P. Gohil. “Ant colony optimized routing for mobile adhoc networks (MANETs)”. International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN), Vol. 2, No. 3, pp. 8-13, 2012.

4. C.Murthy, B.Manoj. "Ad hoc wireless networks: Architectures and protocols". Pearson education, 2004.

5. M. Abolhasan, T. Wysocki, and E. Dutkiewicz. "A review of routing protocols for mobile ad hoc networks". Ad hoc networks, Vol. 2, No. 1, pp. 1-22, 2004.

6. A. Boukerche, B. Turgut, N. Aydin, M. Ahmad, L. Bölöni, D. Turgut. "Routing protocols in ad hoc networks: A survey". Computer Networks, Vol. 55, No. 13, pp. 3032-3080, 2011.

7. S. Mohseni, R. Hassan, A. Patel, R. Razali. "Comparative review study of reactive and proactive routing protocols in MANETs", 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST), held at Dubai, pp. 304-309, Apr. 2010.

8. C. Perkins, E. Royer, S. Das. "RFC 3561-ad hoc on-demand distance vector (AODV) routing", Internet RFCs, pp. 1-38, July 2003.

9. C. Perkins and E. Royer. "Ad Hoc On-Demand Distance Vector Routing". Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), held at New Orleans, Louisiana, pp. 90-100, Feb. 1999.

10. M. Marina, S. Das. "On-demand multipath distance vector routing in ad hoc networks". Ninth IEEE International Conference on Network Protocols, held at Riverside, California, USA, pp. 14-23, Nov. 2001.

11. F. Ducatelle, G. Caro, L. Gambardella. "Principles and applications of swarm intelligence for adaptive routing in telecommunications networks". Swarm Intelligence, Vol. 4, No. 3, pp. 173-198, 2010.

12. G. Caro, F. Ducatelle, and L. Gambardella. "Swarm intelligence for routing in mobile ad hoc networks". Proceedings of the IEEE Swarm Intelligence Symposium, held at Pasadena, USA, pp. 76-83, June 2005.

13. M. Dorigo, M. Birattari, T. Stützle. "Ant colony optimization". IEEE Computational Intelligence Magazine, Vol. 1, No. 4, pp. 28-39, 2006.

14. S. Goss, S. Aron, J. Deneubourg, J. Pasteels. "Self-organized shortcuts in the Argentine ant". Naturwissenschaften, Vol. 76, No. 12, pp. 579-581, 1989.

15. X. Yang. "Nature-inspired optimization algorithms". Elsevier, 2014.

16. X. Yang. "Firefly algorithms for multimodal optimisation". 5th International Symposium by Springer Berlin Heidelberg stochastic algorithms on foundations and applications, held at Japan, Vol. 5792, pp. 169–78, 2009.

17. M. Golshahi, M. Mosleh, M. Kheyrandish. "Implementing an ACO routing algorithm for AD-HOC networks". IEEE International Conference on Advanced Computer Theory and Engineering, ICACTE'08, held at Phuket, Thailand, pp. 143-147, Dec. 2008.

18. Y. Ko, N. Vaidya. "Location-Aided Routing (LAR) in mobile ad hoc networks". Wireless Networks, Vol. 6, No. 4, pp. 307-321, 2000.

19. C. Daniel, A. Loureiro. "A gps/ant-like routing algorithm for ad hoc networks". IEEE Wireless Communications and Networking Conference, WCNC, held at Chicago, IL, USA, Vol. 3, pp. 1232-1236, Sept. 2000.

20. S. Marwaha, C. Tham, and D. Srinivasan. "A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks". 10th IEEE

International Conference on Networks, ICON, held at Grand Copthorne Waterfront, Singapore, pp. 311-316, Aug. 2002.

21. C. Caro, M. Dorigo. “AntNet: A mobile agents approach to adaptive routing”. Ninth Dutch Conference on Artificial Intelligence (NAIC '97), technical report IRIDIA/97-12, held at Antwerpen, Belgium, Nov. 1997.

22. J. S. Baras and H. Mehta. “A Probabilistic Emergent Routing Algorithm (PERA) for Mobile Ad Hoc Networks". Proceedings of WiOpt ’03: Modeling and Optimization in Mobile, AdHoc and Wireless Networks, held at Sophia-Antipolis, France, Mar. 2003.

23. Z. Yuan-Yuan, H. Yan-Xiang. "Ant routing algorithm for mobile ad-hoc networks based on adaptive improvement". Proceedings of the IEEE International Conference on Wireless Communications, Networking and Mobile Computing, held at China, Vol. 2, pp. 678-681, Sept. 2005.

24. D. Kumar, S. Bhuvaneswaran. "Proposal on multi agent ants based routing algorithm for mobile ad-hoc networks". International Journal of Computer Science and Network Security, IJCSNS, Vol. 7, No. 6, pp. 260-268, 2007.

25. J. Wang, E. Osagie, P. Thulasiraman, R. Thulasiram. "HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network". Ad Hoc Networks, Vol. 7, No. 4, pp. 690-705, 2009.

26. E. Osagie , P. Thulasiraman, R. Thulasiram. "PACONET: imProved ant colony optimization routing algorithm for mobile ad hoc networks". 22nd IEEE International Conference on Advanced Information Networking and Applications, AINA 2008, held at GinoWan, Okinawa, Japan, pp. 204-211, Mar. 2008.

27. D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga "A comprehensive survey: artificial bee colony (ABC) algorithm and applications". Artificial intelligence Review, Vol. 42, No. 1, pp. 21-57, 2014.

28. M. Younes, M. Maamar. "Improvement of the ACO meta-heuristic by using the method Artificial Bees Colony (ABC)". Przegląd Elektrotechniczny, Vol. 88, No. 5b, pp. 174-178, 2012.

29. P. Hoolimath, M. Kiran, G. Reddy. "Optimized Termite: A bio-inspired routing algorithm for MANET's". International Conference on Signal Processing and Communications (SPCOM), held at Indian Institute Of Science, Bangalore, pp. 1-5, July 2012.
30. S. Suguna, S. Maheswari. "Comparative Analysis of Bee-Ant Colony Optimized Routing (BACOR) with Existing Routing Protocols for Scalable Mobile Ad Hoc Networks (MANETs) based on Pause Time". International Journal of Computer Science and Network Security, IJCSNS, Vol. 12, No. 4, pp. 9-14, 2012.
31. S. Sharvani, A. Ananth, T. Rangaswamy. "Efficient stagnation avoidance for MANETs with local repair strategy using ant colony optimization". International Journal of Distributed and Parallel Systems (IJDPS), Vol. 3, No. 5, pp. 123-137, 2012.
32. M. Günes, U. Sorges, I. Bouazizi. "ARA-the ant-colony based routing algorithm for MANETs". Proceedings of the IEEE International Conference on Parallel Processing Workshops, held at Vancouver, B.C., Canada, pp. 79-85, 2002.
33. G. Caro, F. Ducatelle, L. Gambardella, M. Dorigo. "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks". European Transactions on Telecommunications, Vol. 16, No. 5, pp. 443-455, 2005.
34. S. Gudakahriz, S. Jamali, E. Zeinali. "Nisr: A nature inspired scalable routing protocol for mobile ad hoc networks". International Journal of Computer Science Engineering and Technology, Vol. 1, No. 4, pp. 180-184, 2011.
35. X. Yang, " Introduction to computational mathematics". World Scientific, 2008.
36. M. Yu, A. Malvankar, W. Su, S. Foo. "A link availability-based QoS-aware routing protocol for mobile ad hoc sensor networks". Computer Communications, Vol. 30, No. 18, pp. 3823-3831, 2007.
37. http://en.wikipedia.org/wiki/Ns_ (simulator) [last accessed on 11-03-2015].