Chapter 2

BACKGROUND

Earliest MANETs were called “packet radio” networks, this project sponsored by Defense Advanced Research Projects Agency (DARPA) in 1970s. Current MANETs are designed primary for military utility; examples include JTRS (Joint Tactical Radio System) and NTDR (Near-Term Digital Radio). Since their emergence in the 70s, wireless networks have become increasingly popular in the computing industry. This is particularly true within the past decade, the wireless networks being adapted to enable mobility.

There are currently two variations of mobile wireless networks. The first is known as the infrastructure network (i.e., a network with fixed and wired gateways). The bridges for these networks are known as base stations. A mobile unit within these networks connects to, and communicates with, the nearest base station that is within its communication radius. As the mobile travels out of range of one base station and into the range of another, a “handoff” occurs from the old base station to the new, and the mobile is able to continue communication seamlessly throughout the network. Typical applications of this type of network include office wireless local area networks (WLANs).

The second type of mobile wireless network is the infrastructure less mobile network, commonly known as an ad hoc network. Infrastructures less networks have no fixed routers; all nodes are capable of movement and can be connected dynamically in an arbitrary manner. Nodes of these networks function as routers which discover and maintain routes to other nodes in the network. Example applications of ad hoc networks are emergency search-and-rescue operations, meetings or conventions in which persons wish to quickly share information.

Since the advent of Defense Advanced Research Projects Agency (DARPA) packet radio networks in the early 1970s, numerous routing protocols have been developed for ad hoc mobile networks. Such protocols must deal with the typical limitations of these networks, which include high power consumption, low bandwidth, and high error rates.

As shown in Figure 2.1, these routing protocols may generally be categorized as: Table-driven (proactive) and Source-initiated (reactive).

2.1 Routing in Mobile Ad Hoc Network

The lack of a backbone infrastructure coupled with the fact that mobile Ad Hoc networks change their topology frequently and without prior notice makes packet routing in ad-hoc networks a challenging task. The suggested approaches for routing can be divided into ***single path and multi path routing.***

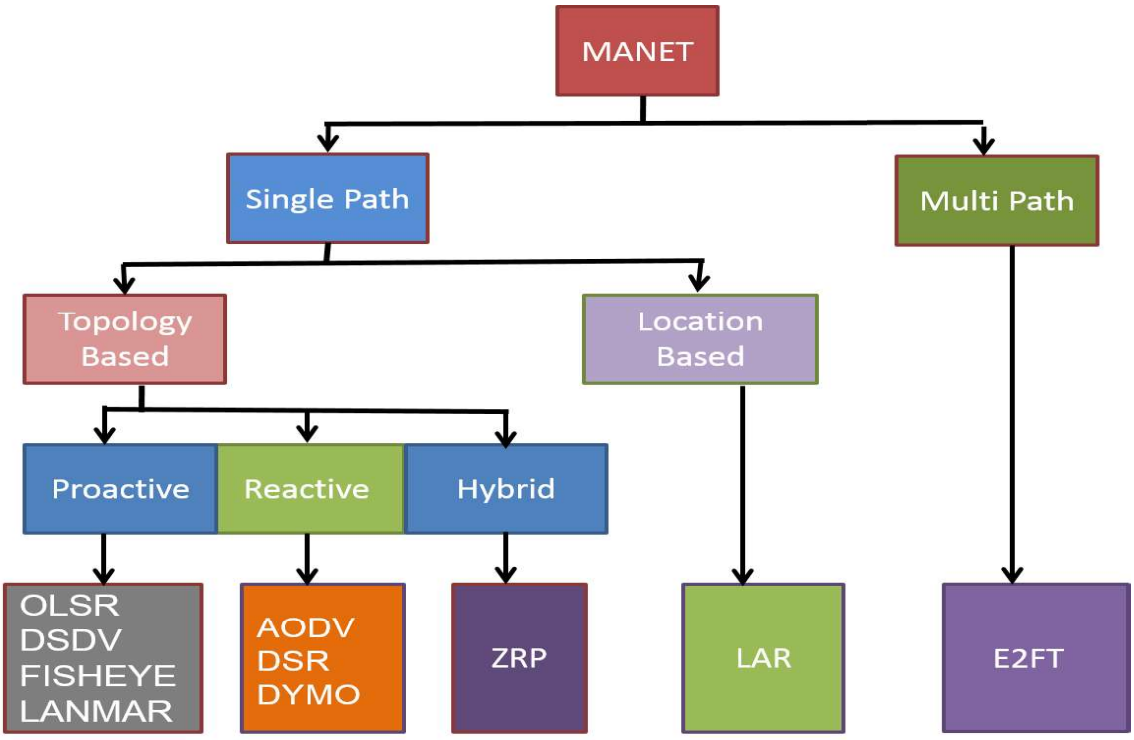


Figure 2.1 Classification of MANETs routing protocols

Single path routing learns routes and selects the best route to destination. It can be further divided into topology-based and location-based routing protocols.

Topology based routing protocols use the information about the links that exist in the network to perform packet forwarding. They can be further divided into **proactive, reactive, and hybrid** approaches.

Proactive algorithms employ classical routing strategies such as distance-vector routing (e.g. DSDV) or link-state routing (e.g. OLSR). They maintain routing information

about the available paths in the network even if these paths are not currently used. The main drawback of these approaches is that the maintenance of unused paths may occupy a significant part of the available bandwidth if the topology of the network changes frequently.

In response to this observation, **reactive** routing protocols were developed (e.g. DSR and AODV). Reactive routing protocols maintain only the routes that are currently in use, thereby reducing the burden on the network when only a small subset of all available routes is in use at any time. However, they still have some inherent limitations.

First, since routes are only maintained while in use, it is typically required to perform a route discovery before packets can be exchanged between communication peers. This leads to a delay for the first packet to be transmitted. Second, even though route maintenance for reactive algorithms is restricted to the routes currently in use, it may still generate a significant amount of network traffic when the topology of the network changes frequently. Finally, packets en-route to the destination are likely to be lost if the route to the destination changes.

Hybrid routing protocols such as ZRP combine local proactive routing and global reactive routing in order to achieve a higher level of efficiency and scalability. However, even a combination of both strategies still needs to maintain at least those network paths that are currently in use, limiting the amount of topological changes that can be tolerated within a given amount of time.

Location based routing algorithms (e.g. LAR) eliminate some of the limitations of topology-based routing by using additional information. They require that information about the physical position of the participating nodes be available. Commonly, each node determines its own position through the use of GPS or some other type of positioning service. A location service is used by the sender of a packet to determine the position of the destination and to include it in the packet's destination address. The routing decision at each node is then based on the destination's position contained in the packet and the position of the forwarding node's neighbours. Position-based routing thus does not require the establishment or maintenance of routes. The nodes have neither to store routing tables nor to transmit messages to keep routing tables up to

date. As a further advantage, position-based routing supports the delivery of packets to all nodes in a given geographic region in a natural way. This type of service is called geocasting.

Multi path routing protocols learn routes and can select more than one path to destination. Hence they are more capable of load balancing, which eventually leads to fault tolerance.

Regardless of the approach to routing, a routing protocol should be able to automatically recover from any problem in a finite amount of time without human intervention. Conventional routing protocols are designed for non-moving infrastructures and assume that routes are bidirectional, which is not always the case for ad-hoc networks. Identification of mobile terminals and correct routing of packets to and from each terminal while moving are certainly challenging.

2.1.1 Routing Protocols studied

In this section we discuss the routing algorithms, which we have considered for our simulations.

2.1.1.1 OLSR [36]

It operates as a table driven, proactive protocol, i.e., exchanges topology information with other nodes of the network regularly. Each node selects a set of its neighbor nodes as "multipoint relays" (MPR). In OLSR, only nodes, selected as such MPRs, are responsible for forwarding control traffic, intended for diffusion into the entire network. MPRs provide an efficient mechanism for flooding control traffic by reducing the number of transmissions required.

Nodes, selected as MPRs, also have a special responsibility when declaring link state information in the network. Indeed, the only requirement for OLSR to provide shortest path routes to all destinations is that MPR nodes declare link-state information for their MPR selectors. Additional available link-state information may be utilized, e.g., for redundancy. Nodes which have been selected as multipoint relays by some neighbour node(s) announce this information periodically in their control messages. Thereby a

node announces to the network, that it has reachability to the nodes which have selected it as an MPR. In route calculation, the MPRs are used to form the route from a given node to any destination in the network. Furthermore, the protocol uses the MPRs to facilitate efficient flooding of control messages in the network.

A node selects MPRs from among its one hop neighbours with "symmetric", i.e., bi-directional, linkages. Therefore, selecting the route through MPRs automatically avoids the problems associated with data packet transfer over unidirectional links (such as the problem of not getting link-layer acknowledgments for data packets at each hop, for link-layers employing this technique for unicast traffic).

OLSR is developed to work independently from other protocols. Likewise, OLSR makes no assumptions about the underlying link-layer. OLSR inherits the concept of forwarding and relaying from HIPERLAN (a MAC layer protocol). The protocol is developed in the IPANEMA project (part of the Euclid program) and in the PRIMA project (part of the RNRT program).

OLSR is modularized into a "core" of functionality, which is always required, for the protocol to operate, and a set of auxiliary functions. The core specifies, in its own right, a protocol able to provide routing in a stand-alone MANET. Each auxiliary function provides additional functionality, which may be applicable in specific scenarios, e.g., in case a node is providing connectivity between the MANET and another routing domain.

The purpose of dividing the functioning of OLSR into core functionality and a set of auxiliary functions is to provide a simple and easy-to-comprehend protocol, and to provide a way of only adding complexity where specific additional functionality is required. Specifically, the core is made up from the following components:

Packet Format and Forwarding: A universal specification of the packet format and an optimized flooding mechanism serves as the transport mechanism for all OLSR control traffic.

Link Sensing: Link Sensing is accomplished through periodic emission of HELLO messages over the interfaces through which connectivity is checked. A separate HELLO message is generated for each interface and emitted. If sufficient information is

provided by the link-layer, this may be utilized to populate the local link set instead of HELLO message exchange.

Neighbour detection: Given a network with only single interface nodes, a node may deduct the neighbour set directly from the information exchanged as part of link sensing: the "main address" of a single interface node is, by definition, the address of the only interface on that node. In a network with multiple interface nodes, additional information is required in order to map interface addresses to main addresses (and, thereby, to nodes). This additional information is acquired through multiple interface declaration (MID) messages.

MPR Selection and MPR Signalling: The objective of MPR selection is for a node to select a subset of its neighbours such that a broadcast message, retransmitted by these selected neighbours, will be received by all nodes 2 hops away. The MPR set of a node is computed such that it, for each interface, satisfies this condition. The information required to perform this calculation is acquired through the periodic exchange of HELLO messages.

Topology Control Message Diffusion: Topology Control messages are diffused with the purpose of providing each node in the network with sufficient link-state information to allow route calculation.

2.1.1.2 DSDV: Destination-Sequenced Distance Vector Protocol [37]

The Destination-Sequenced Distance Vector (DSDV) protocol is a table-driven routing protocol based on the improved version of classical Bellman-Ford routing algorithm. DSDV is based on the Routing Information Protocol (RIP). With RIP, a node holds a routing table containing all the possible destinations within the network and the number of hops to each destination. DSDV is also based on distance vector routing and thus uses bidirectional links. A limitation of DSDV is that it provides only one route for a source/destination pair. DSDV requires each node to periodically broadcast routing updates.

The key advantage of DSDV over traditional distance vector protocols is that it guarantees loop-freedom**.** Each DSDV node maintains a routing table listing the “next

hop" for each reachable destination. DSDV tags each route with a sequence number and considers a route R1 more favourable than R2 if R1 has a greater sequence number, or if the two routes have equal sequence numbers but R has a lower metric. Each node in the network advertises a monotonically increasing even sequence number for itself. When a node B decides that its route to a destination D has broken, it advertises the route to D with an infinite metric and a sequence number one greater than its sequence number for the route that has broken (making an odd sequence number). This causes any node 'A' routing packets through 'B' to incorporate the infinite-metric route into its routing table until node 'A' hears a route to 'D' with a higher sequence number.

The structure of the routing table for this protocol is simple. Each table entry has a sequence number that is incremented every time a node sends an updated message. Routing tables are periodically updated when the topology of the network changes and are propagated throughout the network to keep consistent information throughout the network.

Each DSDV node maintains two routing tables: one for forwarding packets and one for advertising incremental routing packets. The routing information sent periodically by a node contains a new sequence number, the destination address, the number of hops to the destination node, and the sequence number of the destination.

2.1.1.3 FSR: Fisheye State Routing Protocol [38]

It is an implicit hierarchical routing protocol. It uses the "fisheye" technique. The eye of a fish captures with high detail the pixels near the focal point. The detail decreases as the distance from the focal point increases. In routing, the fisheye approach translates to maintaining accurate distance and path quality information about the immediate neighborhood of a node, with progressively less detail as the distance increases.

FSR is functionally similar to Link State Routing in that it maintains a topology map at each node. The key difference is the way in which routing information is disseminated. In LS, link state packets are generated and flooded into the network whenever a node detects a topology change. In FSR, link state packets are not flooded. Instead, nodes maintain a link state table based on the up-to-date information received from neighboring nodes, and periodically exchange it with their local neighbors only (no

flooding). Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence numbers. In FSR link states rather than distance vectors are propagated. A full topology map is kept at each node and shortest paths are computed using this map. In a wireless environment, a radio link between mobile nodes may experience frequent disconnects and reconnects. FSR avoids this problem by using periodic, instead of event driven, exchange of the topology map, greatly reducing the control message overhead. When network size grows large, the update message could consume considerable amount of bandwidth, which depends on the update period. In order to reduce the size of update messages without seriously affecting routing accuracy, FSR uses the fisheye technique. The reduction of routing update overhead is obtained by using different exchange periods for different entries in routing table. More precisely, entries corresponding to nodes within the smaller scope are propagated to the neighbors with the highest frequency. A considerable fraction of link state entries are suppressed in a typical update, thus reducing the message size. This strategy produces timely updates from near stations, but creates large latencies from stations afar. However the imprecise knowledge of the best path to a distant destination is compensated by the fact that the route becomes progressively more accurate as the packet gets closer to destination. As the network size grows large, a "graded" frequency update plan must be used across multiple scopes to keep the overhead low.

The entire topology table is exchanged among neighbors. Clearly, this consumes a considerable amount of bandwidth when network size becomes large. Through updating link state information with different frequencies depending on the scope distance, FSR scales well to large network size and keeps overhead low without compromising route computation accuracy when the destination is near. By retaining a routing entry for each destination, FSR avoids the extra work of "finding" the destination and thus maintains low single packet transmission latency. As mobility increases, routes to remote destinations become less accurate. However, when a packet approaches its destination, it finds increasingly accurate routing instructions as it enters sectors with a higher refresh rate.

2.1.1.4 LANMAR: Landmark Ad Hoc Routing Protocol [39]

In the original landmark scheme for wired networks, the predefined hierarchical address of each node reflects its position within the hierarchy and helps find a route to it. Each node knows the routes to all the nodes within it hierarchical partition. Moreover, each node knows the routes to various "landmarks" at different hierarchical levels. Packet forwarding is consistent with the landmark hierarchy and the path is gradually refined from top level hierarchy to lower levels as a packet approaches destination. LANMAR borrows the notion of landmarks to keep track of logical subnets. A subnet consists of members who have a commonality of interests and are likely to move as a "group" (e.g., brigade in the battlefield, colleagues in the same organization, or a group of students from same class). A "landmark" node is elected in each subnet. The routing scheme itself is modified version of FSR. The main difference is that the FSR routing table contains "all" nodes in the network, while the LANMAR routing table includes only the nodes within the scope and the landmark nodes. This feature greatly improves scalability by reducing routing table size and update traffic O/H. When a node needs to relay a packet, if the destination is within its neighbor scope, the address is found in the routing table and the packet is forwarded directly. Otherwise, the logical subnet field of the destination is searched and the packet is routed towards the landmark for that logical subnet. The packet however does not need to pass through the landmark. Rather, once the packet gets within the scope of the destination, it is routed to it directly. The routing update exchange in LANMAR routing is similar to FSR. Each node periodically exchanges topology information with its immediate neighbors. In each update, the node sends entries within its fisheye scope. It also piggy-backs a distance vector with size equal to the number of logical subnets and thus landmark nodes. Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence numbers.

2.1.1.5 AODV: Ad Hoc On-Demand Distance Vector Protocol [40]

AODV [25] can be thought of as a combination of both DSR and DSDV. It borrows the basic on-demand mechanism of Route Discovery and Route Maintenance from DSR, plus

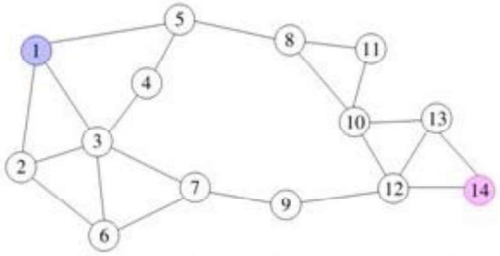
the use of hop-by-hop routing, sequence numbers, and periodic beacons from DSDV. AODV is an on-demand routing protocol, which initiates a route discovery process only when desired by a source node. When a source node S wants to send data packets to a destination node D but cannot find a route in its routing table, it broadcasts a Route Request (RREQ) message to its neighbours, including the last known sequence number for that destination. Its neighbours then rebroadcast the RREQ message to their neighbours if they do not have a fresh enough route to the destination node. (A fresh enough route is a valid route entry for the destination node whose associated sequence number is equal to or greater than that contained in the RREQ message.) This process continues until the RREQ message reaches the destination node or an intermediate node that has a fresh enough route. Every node has its own sequence number and RREQ ID1. AODV uses sequence numbers to guarantee that all routes are loop-free and contain the most recent routing information. RREQ ID in conjunction with source IP address uniquely identifies a particular RREQ message. The destination node or an intermediate node only accepts the first copy of a RREQ message, and drops the duplicated copies of the same RREQ message. Each node that forwards the ROUTE REQUEST creates a *reverse route* for itself back to node S; after accepting a RREQ message, the destination or intermediate node updates its reverse route to the source node using the neighbour from which it receives the RREQ message. The reverse route will be used to send the corresponding Route Reply (RREP) message to the source node – when the ROUTE REQUEST reaches a node with a route to D, that node generates a ROUTE REPLY that contains the number of hops necessary to reach D and the sequence number for D most recently seen by the node generating the REPLY. Meanwhile, it updates the sequence number of the source node in its routing table to the maximum of the one in its routing table and the one in the RREQ message. When the source or an intermediate node receives a RREP message, it updates its *forward route* to the destination node using the neighbour from which it receives the RREP message. It also updates the sequence number of the destination node in its routing table to the maximum of the one in its routing table and the one in the RREP message. A Route Reply Acknowledgement (RREP-ACK) message is used to acknowledge receipt of a RREP message. The state created in each node along the path from S to D is hop-by-hop state; that is, each node remembers only the next hop and not the entire route, as would be done in source routing.

In order to maintain routes, AODV normally requires that each node periodically transmit a HELLO message, with a default rate of once per second. Failure to receive three consecutive HELLO messages from a neighbour is taken as an indication that the link to the neighbour in question is down. Alternatively, the AODV specification briefly suggests that a node may use physical layer or link layer methods to detect link breakages to nodes that it considers neighbours. When a link goes down, any upstream node that has recently forwarded packets to a destination using that link is notified via an UNSOLICITED ROUTE REPLY containing an infinite metric for that destination. Upon receipt of such a ROUTE REPLY, a node must acquire a new route to the destination using Route Discovery as described above.

Route maintenance is done with Route Error (RERR) messages. If a node detects a link break in an active route, it sends out a RERR message to its upstream neighbours that use it as the next hop in the broken route. When a node receives a RERR message from its neighbour, it further forwards the RERR message to its upstream neighbours.

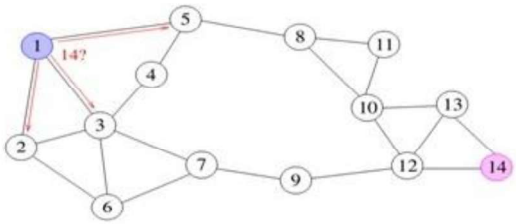
AODV is a stateless protocol; the source node or an intermediate node updates its routing table if it receives a RREP message, regardless of whether it has sent or forwarded a corresponding RREQ message before. If it cannot find the next hop in the reverse routing table, it simply drops the RREP message. Otherwise, it unicasts the RREP message to the next hop in the reverse route. A general working structure of protocol is given below:

Fig. A AODV (RFC3561)



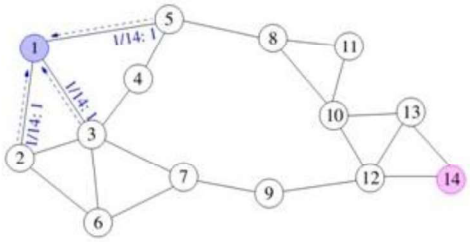
- on-demand routing protocol
- node 1 → 14

Fig. B AODV (RREQ)



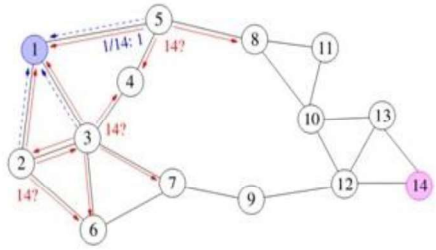
- broadcast flood route request message
  - (broadcast traffic in red)

Fig. C AODV (RREQ)



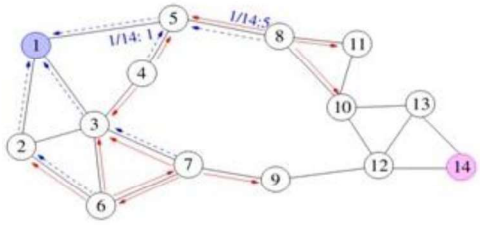
- node from which RREQ was received defines a reverse route to the source
  - ("reverse routing table entries" blue)

Fig. D AODV (RREQ)



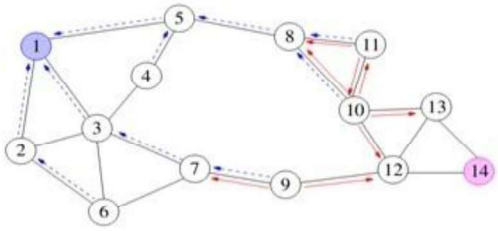
- route request is flooded through the network
  - reverse routing table entries (blue arrows)

Fig. E AODV (RREQ)



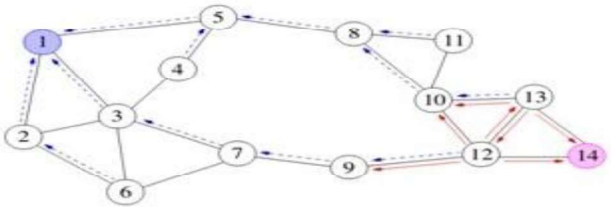
- unreliable communication
- destination managed sequence number, ID prevent looping

Fig. F AODV (RREQ)



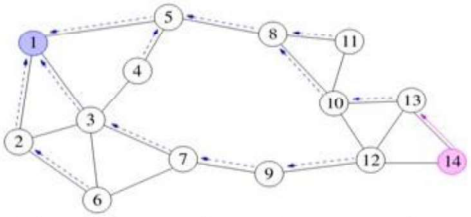
- flooding is expensive
- broadcast collision problem

Fig. G AODV (RREQ)



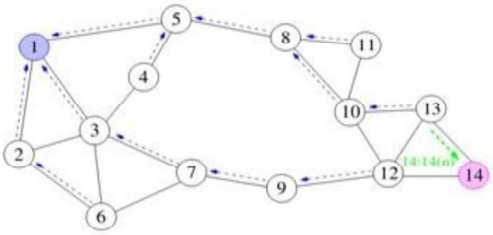
- route request arrives at the destination
- two routes are discovered

Fig. H AODV (RREP)



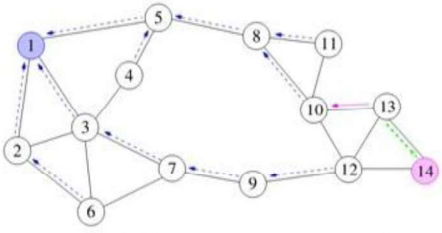
- destination sends route reply (set sequence number)
  - (unicast reply in magenta)

Fig. I AODV (RREP)



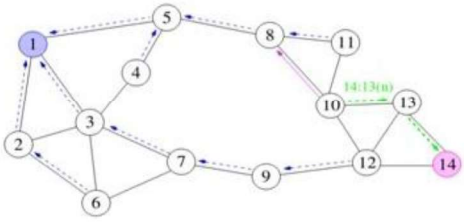
- routing table now contains forward route to the destination
  - ("reverse routing table entries" in blue)

Fig. J AODV (RREP)



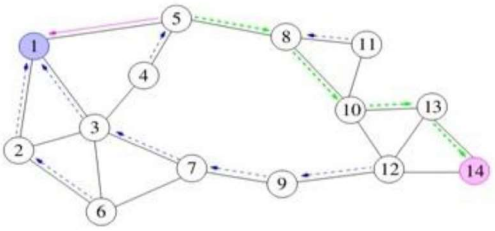
- route reply follows reverse route back to the source

Fig. K AODV (RREP)



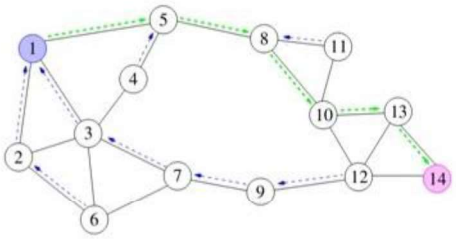
- setting the forward routing table entries along the way

Fig. L AODV (RREP)



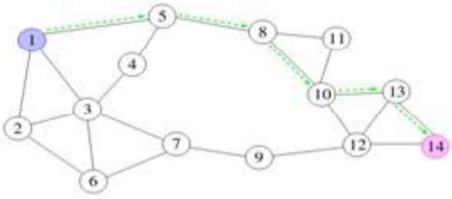
▪ route reply reaches the source

Fig. M AODV (RREP)



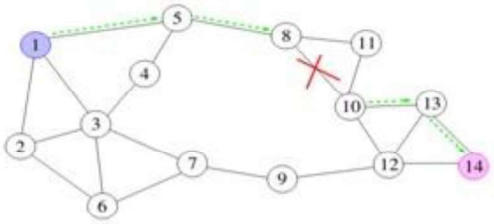
▪ source adopts destination sequence number

Fig. N AODV (RREP)



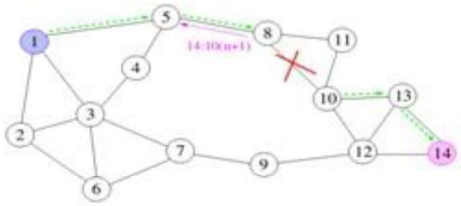
▪ traffic flows along the forward route
▪ forward route is refreshed, reverse routes time out

Fig. O AODV (RERR)



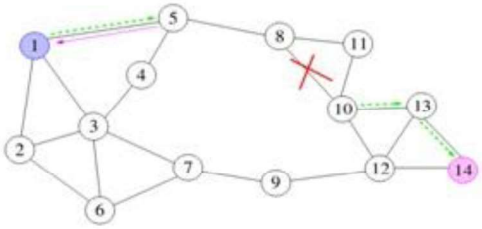
▪ link failure detection

Fig. P AODV (RERR)



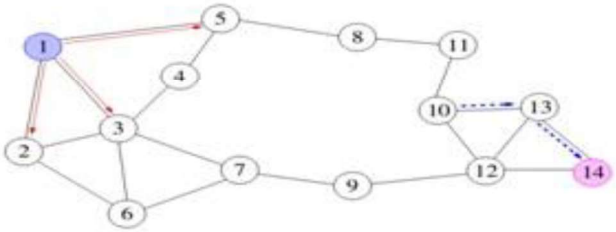
▪ return error message to the source (increment sequence number)

Fig. Q AODV (RERR)



▪ source receives route error

Fig. R AODV (RREQ)



▪ re-initiates route discovery process

Figure 2.2[A-R]: AODV working Structure

In general, a node may update the sequence numbers in its routing table whenever it receives RREQ, RREP, RERR, or RREP-ACK messages from its neighbours.

2.1.1.6 DSR: Dynamic Source Routing Protocol [41]

It uses source routing rather than hop-by-hop routing, with each packet to be routed carrying in its header the complete, ordered list of nodes through which the packet must pass. The key advantage of source routing is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. This fact, coupled with the on-demand nature of the protocol, eliminates the need for the periodic route advertisement and neighbour detection packets present in other protocols.

The DSR protocol consists of two mechanisms: Route Discovery and Route Maintenance.

Route Discovery is the mechanism by which a node S wishing to send a packet to a destination D obtains a source route to D. To perform a Route Discovery, the source node S broadcasts a ROUTE REQUEST packet that is flooded through the network in a controlled manner and is answered by a ROUTE REPLY packet from either the destination node or another node that knows a route to the destination. To reduce the cost of Route Discovery, each node maintains a cache of source routes it has learned or overheard, which it uses to limit the frequency and propagation of ROUTE REQUESTs.

Route Maintenance is the mechanism by which a packet's sender S detects if the network topology has changed such that it can no longer use its route to the destination D because two nodes listed in the route have moved out of range of each other. When Route Maintenance indicates a source route is broken, S is notified with a ROUTE ERROR packet. The sender S can then attempt to use any other route to D already in its cache or can invoke Route Discovery again to find a new route.

2.1.1.7 DYMO: Dynamic MANET On-demand Protocol [42]

It enables dynamic, reactive, multi-hop routing between participating nodes wishing to communicate. The basic operations of the protocol are route discovery

and management. During route discovery the originating node causes dissemination of a Routing Element (RE) throughout the network to find the target node. During dissemination each intermediate node creates a route to the originating node. When the target node receives the RE it responds with RE unicast toward originating node. During propagation each node creates a route to the target node. When the originating node is reached routes have been established between the originating node and the target node in both directions.

In order to react quickly to changes in the network topology nodes should maintain their routes and monitor their links. When a packet is received for a route that is no longer available the source of the packet should be notified. A Route Error (RERR) is sent to the packet source to indicate the current route is broken. Once the source receives the RERR, it will re-initiate route discovery if it still has packets to deliver.

In order to enable extension of the base specification, DYMO defines the handling of unsupported extensions. By defining default handling, future extensions are handled in a predetermined understood fashion. DYMO uses sequence numbers to ensure loop freedom. The route table has a list of route entries, where each route entry has the following fields:

- Address: the IP address of the destination node.
- SeqNum: the sequence number of the destination known by the node.
- NextHop: the IP address of the next hop on the path towards the destination.
- Broken: a flag indicating whether the route entry is valid.
- Dist: a metric indicating the distance traversed from the node to reach the destination. Prefix: indicates that the associated address is a network address, rather than a host address.

2.1.1.8 ZRP: Zone Routing Protocol [43]

It aims to combine the best properties of both reactive and proactive protocol. It is a hybrid reactive/proactive routing protocol. ZRP reduces the proactive scope to a zone centered on each node. In a limited zone, the maintenance of routing information is

easier. Further, the amount of routing information that is never used is minimized. Still, nodes farther away can be reached with reactive routing. Since all nodes proactively store local routing information, route requests can be more efficiently performed without querying all the network nodes.

Despite the use of zones, ZRP has a flat view over the network. In this way, the organizational overhead related to hierarchical protocols can be avoided. Hierarchical routing protocols depend on the strategic assignment of gateways or landmarks, so that every node can access all levels, especially the top level. Nodes belonging to different subnets must send their communication to a subnet that is common to both nodes. This may congest parts of the network.

A routing zone is defined for each node separately, and the zones of neighboring nodes overlap. The routing zone has a radius expressed in hops. The zone thus includes the nodes, whose distance from the node in question is at most the radius in hops.

The nodes of a zone are divided into peripheral nodes and interior nodes. Peripheral nodes are nodes whose minimum distance to the central node is exactly equal to the zone radius. The nodes whose minimum distance is less than the radius are interior nodes. The number of nodes in the routing zone can be regulated by adjusting the transmission power of the nodes. Lowering the power reduces the number of nodes within direct reach and vice versa. The number of neighboring nodes should be sufficient to provide adequate reachability and redundancy. On the other hand, a too large coverage results in many zone members and the update traffic becomes excessive. Further, large transmission coverage adds to the probability of local contention.

ZRP refers to the locally proactive routing component as the IntrA-zone Routing Protocol (IARP). The globally reactive routing component is named IntEr-zone Routing Protocol (IERP). IERP and IARP are not specific routing protocols. Instead, IARP is a family of limited-depth, proactive link-state routing protocols. IARP maintains routing information for nodes that are within the routing zone of the node. Correspondingly, IERP is a family of reactive routing protocols that offer enhanced route discovery and route maintenance services based on local connectivity monitored by IARP.

The fact that the topology of the local zone of each node is known can be used to reduce traffic when global route discovery is needed. Instead of broadcasting packets, ZRP uses a concept called bordercasting. Bordercasting utilizes the topology information provided by IARP to direct query request to the border of the zone. The bordercast packet delivery service is provided by the Bordercast Resolution Protocol (BRP). BRP uses a map of an extended routing zone to construct bordercast trees for the query packets. Alternatively, it uses source routing based on the normal routing zone. By employing query control mechanisms, route requests can be directed away from areas of the network that already have been covered. In order to detect new neighbor nodes and link failures, the ZRP relies on a Neighbor Discovery Protocol (NDP) provided by the Media Access Control (MAC) layer. NDP transmits "HELLO" beacons at regular intervals. Upon receiving a beacon, the neighbor table is updated. Neighbors, for which no beacon has been received within a specified time, are removed from the table. Route updates are triggered by NDP, which notifies IARP when the neighbor table is updated. IERP uses the routing table of IARP to respond to route queries. IERP forwards queries with BRP. BRP uses the routing table of IARP to guide route queries away from the query source.

2.1.1.9 LAR: Location Aided Routing Protocol [44]

It is a geographic routing protocol or position-based routing protocol. This protocol assumes that source node sends data to destination node using the information of co-ordinate geographic location of the destination. The network address becomes unusable in this case but the performance is seen to improve. The overhead which is generated by route discovery packets accounts for decreased traffic overhead. Global Positioning System (GPS) can be used in calculating the location of a node.

The unique characteristics of location based protocols are:

i. Each node determines its own location.

ii. Source knows the location of destination.

Due to this property, packets can be successfully routed to destination, removing the necessity of knowledge of network topology. LAR belongs to category of reactive

protocol and is built on DSR (Dynamic Source Routing protocol). In DSR protocol the entire network is flooded by RREQ packet, when route to a specific node is not found. This flooding is minimized in LAR by using the location information. The packets are flooded in a forwarding zone called the Request Zone, which in itself is based on location information. Two techniques were proposed to determine whether a particular node is a candidate of the request zone. These techniques are LAR Box and LAR Step Protocol.

LAR Box Protocol

Here, the candidature of a node is determined by using the location information of the node and expected zone for the node which initiates the flooding. The Expected Zone is determined by following information:

• Most recent location of the node.

• Time at which this location is calculated.

• Average velocity of the node.

• Current time.

If a node is found to be in the request zone, it forwards the route request packet using the same technique of finding the zone flooding the packet.

LAR Step Protocol

Here, the candidature of a node is determined by finding the distance between the node and destination. If this distance is smaller than the previously calculated distance for the node which wants to flood data then its candidature is approved. For e.g. let us suppose that distance between node A (the node which sends the RREQ packet) and node B is Dx, and the distance between node C (the node which received the RREQ packet) and node B is Dy. Now, the node C will forward the RREQ packet if Dy<=Dx.

2.1.1.10 E2FT: End-to-end Estimation-based Fault Tolerant Routing Algorithm [45]

The existence of faulty nodes in ad hoc networks can significantly affect the packet delivery rate of the routing protocols. The goal of designing a fault tolerant routing algorithm is to provide certain packet delivery rate guarantee in the presence of faulty nodes. To achieve this goal, the routing algorithms explore network redundancy through multipath routing. In multipath routing, duplicate packets are sent along the multiple paths between the source and the destination. The packet is regarded as successfully delivered, if one copy of the packets is received. Without the knowledge of faulty nodes, employing multipath routing blindly can introduce extremely high overhead into the network. Thus, the most critical question that needs to be addressed by a fault tolerant routing algorithm is how to select the paths so that packet deliver rate can be guaranteed through minimum packet duplication. The source node makes centralized decision on path selection. Furthermore, it is assumed that the source knows multiple paths between itself and the destination through route discovery procedure. It consists of two processes: route estimation and route selection.

Route estimation: It estimates the delivery probability of the available paths. The source node estimates the probability of path p by sending data packets along p, and measuring its packet delivery rate. The destination helps the source with the estimation procedure by sending feedback with information how many packets were received. The accuracy of the estimation result depends on the number of packets used for estimation. With this estimation method, the past behavior of the path p has the same weight as its current behavior. Thus, this estimation needs the path p behave consistently in packet delivery. The estimation results of different paths may be based on different n values. Thus they have different accuracies. To facilitate the path selection, we need unified path delivery rate estimation so that paths can be selected fairly.

Route selection: It selects the set of paths on a fair basis. Since the accuracy of the path estimations is increased through iterations, the route selection also refines its path selection progressively with the increasingly accurate estimation results. With increasingly accurate estimation results, E2FT refines its path selection through two procedures – confirmation and dropping.

- Confirmation is a procedure that, through "accurate enough" estimation, a single path is verified to be able to deliver the packets with the expected rate, alone.
- Dropping is a greedy decision procedure that removes the unnecessary paths. This procedure picks a path with minimum α-estimation.

2.2 Mobility Models

It is designed to describe the movement pattern of mobile users, and how their location, velocity and acceleration change over time. It patterns may play a significant role in determining the protocol performance; it is desirable for mobility models to emulate the movement pattern of targeted real life applications in a reasonable way. In this section, we describe the mobility models used in our studies. The suggested approaches for mobility model are outlined below.

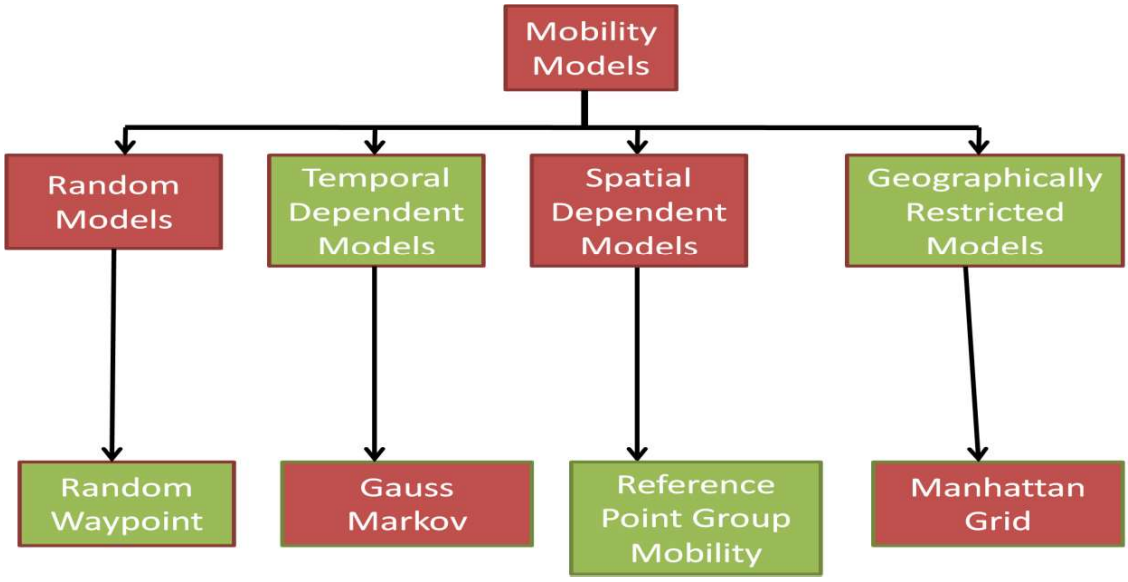


Figure 2.3: Classification of mobility models

Random models: Here the destination is chosen randomly, with randomly selected velocity e.g. Random waypoint mobility model.

Temporal dependent models: Here, the movement of a mobile node is likely to be affected by its movement history e.g. Gauss Markov model.

Spatial dependent models: Here, the mobile nodes tend to travel in a correlated manner e.g. Reference point group mobility model (RPGM).

Geographically restricted models: Here, the movement of nodes is bounded by streets, freeways or obstacles e.g. Manhattan Grid.

An exhaustive survey of mobility models was done by F. Bai et.al [46] and T. Camp et.al [47].

2.2.1 Random Waypoint Mobility Model [48]

It was first proposed by Johnson and Maltz [17]. Soon, it became a 'benchmark' mobility model to evaluate the MANET routing protocols, because of its simplicity and wide availability. To generate the node trace of the Random Waypoint model the setdest tool from the CMU Monarch group may be used. This tool is included in the widely used network simulator ns-2.

As the simulation starts, each mobile node randomly selects one location in the simulation field as the destination. It then travels towards this destination with constant velocity chosen uniformly and randomly from [0, V], where the parameter V is the maximum allowable velocity for every mobile node. The velocity and direction of a node are chosen independently of other nodes. Upon reaching the destination, the node stops for a duration defined by the 'pause time' parameter**.** If T =0, this leads to continuous mobility. After this duration, it again chooses another random destination in the simulation field and moves towards it. The whole process is repeated again and again until the simulation ends. Here, V and T are the two key parameters that determine the mobility behavior of nodes. If the V is small and the pause time T is long, the topology of Ad Hoc network becomes relatively stable. On the other hand, if the node moves fast (i.e., is large) and the pause time T is small, the topology is expected to be highly dynamic. Varying these two parameters, especially the V parameter, the Random Waypoint model can generate various mobility scenarios with different levels of nodal speed.

2.2.2 Gauss Markov Model [49]

It was designed to adapt to different levels of randomness via one tuning parameter. Initially each node is assigned a current speed and direction. At fixed intervals of time, n, movement occurs by updating the speed and direction of each node. Specifically, the value of speed and direction at the n^{th} instance is calculated based upon the value of speed and direction at the $(n-1)^{st}$ instance and a random variable using the following equations:

$$sn = \alpha sn - 1 + (1 - \alpha)\bar{s} + \sqrt{(1 - \alpha 2)}s_{x_{n-1}}$$

$$dn = \alpha d_{n-1} + (1 - \alpha)\bar{d} + \sqrt{(1 - \alpha 2)}d_{x_{n-1}}$$

Where, s_n and d_n are the new speed and direction of the node at time interval n; α, where $0 \leq \alpha \leq 1$, is the tuning parameter used to vary the randomness; s and d, are constants representing the mean value of speed and direction as $n \to \infty$; and $s_{xn}-1$ and $d_{xn}-1$ are random variables from a Gaussian distribution. Totally random values (or Brownian motion) are obtained by setting $\alpha = 0$ and linear motion is obtained by setting $\alpha = 1$. Intermediate levels of randomness are obtained by varying the value of α between 0 and 1.

At each time interval the next location is calculated based on the current location, speed, and direction of movement. Specifically, at time interval n, node position is given by the equations:

$$xn = x_{n-1} + s_{n-1}cosd_{n-1}$$

$$yn = y_{n-1} + s_{n-1}sind_{n-1}$$

where (x_n, y_n) and (x_{n-1}, y_{n-1}) are the x and y coordinates of the nodes position at the n^{th} and $(n-1)^{st}$ time intervals, respectively, and s_{n-1} and d_{n-1} are the speed and direction of the node respectively, at the $(n-1)^{st}$ time interval. To ensure that a node does not remain near an edge of the grid for a long period of time, the nodes are forced away from an edge when they move within a certain distance of the edge. This is done by modifying the mean direction variable d in the above direction equation.

2.2.3 Manhattan Grid model [50]

It was proposed to emulate the movement pattern of mobile nodes on streets defined by maps. It can be useful in modeling movement in an urban area where a pervasive computing service between portable devices is provided. The map is composed of a number of horizontal and vertical streets. Each street has two lanes for each direction (north and south direction for vertical streets, east and west for horizontal streets). The mobile node is allowed to move along the grid of horizontal and vertical streets on the map. At an intersection of a horizontal and a vertical street, the mobile node can turn left, right or go straight. This choice is probabilistic: the probability of moving on the same street is 0.5, the probability of turning left is 0.25 and the probability of turning right is 0.25. The velocity of a mobile node at a time slot is dependent on its velocity at the previous time slot. Also, a nodes velocity is restricted by the velocity of the node preceding it on the same lane of the street. The inter-node and intra-node relationships involved are the same as in the Freeway model. Thus, the Manhattan mobility model is also expected to have high spatial dependence and high temporal dependence. It too imposes geographic restrictions on node mobility. However, it differs from the Freeway model in giving a node some freedom to change its direction.

2.2.4 Reference Point Group Mobility model [51]

It represents the random motion of a group of mobile nodes as well as the random motion of each individual node within the group. Group movements are based upon the path travelled by a logical centre for the group. The logical centre for the group is used to calculate group motion via a group motion vector, $\overrightarrow{GM}$. The motion of the group centre completely characterizes the movement of its corresponding group of nodes, including their direction and speed. Individual nodes randomly move about their own pre-defined reference points, whose movements depend on the group movement. As the individual reference points move from time t to t+1, their locations are updated according to the group's logical centre. Once the updated reference points, RP (t+1), are calculated, they are combined with a random motion vector, $\overrightarrow{RM}$, to represent the random motion of each node about its individual reference point.

It was designed to depict scenarios such as an avalanche rescue. During an avalanche rescue, the responding team consisting of human and canine members work cooperatively. The human guides tend to set a general path for the dogs to follow, since they usually know the approximate location of victims. The dogs each create their own "random" paths around the general area chosen by their human counterparts**.** If appropriate group paths are chosen, along with proper initial locations for various groups, many different mobility applications may be represented with the RPGM model.

2.3 Prologue of Network Simulators

Simulation is the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed representing the key characteristics or functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.

Simulation is used in many contexts, such as simulation of technology for performance optimization, safety engineering, testing, training, education etc. It can be used to show the eventual real effects of alternative conditions and courses of action.

2.3.1 Advantages of Simulation

One of the primary advantages of simulators is that they are able to provide users with practical feedback when designing real world systems. This allows the designer to determine the correctness and efficiency of a design before the system is actually constructed. Consequently, the user may explore the merits of alternative designs without actually physically building the systems. By investigating the effects of specific design decisions during the design phase rather than the construction phase, the overall cost of building the system diminishes significantly.

Another benefit of simulators is that they permit system designers to study a problem at several different levels of abstraction. By approaching a system at a higher level of abstraction, the designer is better able to understand the behaviours and interactions of all the high level components within the system and is therefore better equipped to counteract the complexity of the overall system. As the designer better understands the

operation of the higher level components through the use of the simulator, the lower level components may then be designed and subsequently simulated for verification and performance evaluation. The entire system may be built based upon this ``top-down'' technique. Working at a higher level abstraction also facilitates rapid prototyping in which preliminary systems are designed quickly for the purpose of studying the feasibility and practicality of the high-level design.

For our simulation purposes, we have used NS2, Qualnet and GloMoSim simulator, which are described below.

2.3.2 The Network Simulator (ns-2) [52]

Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. It began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns have always included substantial contributions from other researchers, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.

It is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy in this document), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy in this document). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy.

Its goal is to support networking research and education, protocol design, traffic studies, protocol comparison etc and provide a collaborative environment which is freely distributed and open source to share code, protocols, models etc.

It has two Components:

- Ns, the simulator itself
- Nam, the network animator to visualize ns (or other) output

2.3.3 QualNet®[53]

It is a planning, testing and training tool that "mimics" the behaviour of a real communications network. It provides a comprehensive environment for designing protocols, creating and animating network scenarios, and analysing their performance.

It is composed of the following components:

QualNet Architect: A graphical scenario design and visualization tool. In Design mode, you can set up terrain, network connections, subnets, mobility patterns of wireless users, and other functional parameters of network nodes. You can create network models by using intuitive, click and drag operations. You can also customize the protocol stack of any of the nodes. You can also specify the application layer traffic and services that run on the network. In Visualize mode, you can perform in-depth visualization and analysis of a network scenario designed in Design mode. As simulations are running, users can watch packets at various layers flow through the network and view dynamic graphs of critical performance metrics. Real-time statistics are also an option, where you can view dynamic graphs while a network scenario simulation is running.

QualNet Analyzer: A statistical graphing tool that displays hundreds of metrics collected during simulation of a network scenario. You can choose to see pre-designed reports or customize graphs with their own statistics. Multi-experiment reports are also available. All statistics are exportable to spreadsheets in CSV format.

QualNet Packet Tracer: A graphical tool that provides a visual representation of packet trace files generated during the simulation of a network scenario. Trace files are text files in XML format that contain information about packets as they move up and down the protocol stack.

QualNet File Editor: A text editing tool

QualNet Command Line Interface: Command line access to the simulator

It enables users to:

- Design new protocol models
- Optimize new and existing models
- Design large wired and wireless networks using pre-configured or user-designed models
- Analyse the performance of networks and perform what-if analysis to optimize them

The key features of QualNet that enable creating a virtual network environment are:

- Speed: It can support real-time speed to enable software-in-the-loop, network emulation, and human-in-the-loop modelling. Faster speed enables model developers and network designers to run multiple "what-if" analyses by varying model, network, and traffic parameters in a short time.
- Scalability: It can model thousands of nodes by taking advantage of the latest hardware and parallel computing techniques. QualNet can run on cluster, multi-core, and multi-processor systems to model large networks with high fidelity.
- Model Fidelity: It uses highly detailed standards-based implementation of protocol models. It also includes advanced models for the wireless environment to enable more accurate modelling of real-world networks.
- Portability: It runs on a vast array of platforms, including Windows and Linux operating systems, distributed and cluster parallel architectures, and both 32- and 64-bit computing platforms. Users can now develop a protocol model or design a network in QualNet on their desktop or laptop Windows computer and then transfer it to a powerful multi-processor Linux server to run capacity, performance, and scalability analyses.
- Extensibility: It can connect to other hardware and software applications, such as OTB, real networks, and third party visualization software, to greatly enhance the value of the network model.

2.3.4 GloMoSim [54]

It is a library –based sequential and parallel simulator for wireless networks. It is designed as a set of library modules, each of which simulates a specific wireless communication protocol in the protocol stack. The library has been developed using PARSEC; a C based parallel simulation language. New protocols and modules can be programmed and added to the library using this language. PARSEC adopts a message based approach to discrete event simulation: physical processes are modelled by simulation objects called entities, and events are represented by transmission of time stamped messages among corresponding entities.

It has been implemented on both shared memory and distributed memory computers and can be executed using a variety of synchronization protocols. It has a visualization tool that is platform independent because it is coded in Java. This tool allows to debug and verify models and scenarios; stop, resume and step execution; show packet transmissions, show mobility groups in different colours and show statistics.

It aims to develop a modular simulation environment for protocol stacks, which is capable of scaling up to networks with thousands of heterogeneous nodes. If all protocol models obey the strict APIs defined at each layer, it will be feasible to simple swap protocol models at a certain layer without having to modify the models for the remaining layers in the stack.