Simulation-based Performance Evaluation of MANET Backbone Formation Algorithms

by

Khalid Almahrog

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2007

Abstract

As a result of the recent advances in the computation and communications industries, wireless communications-enabled computing devices are ubiquitous nowadays. Even though these devices are introduced to satisfy the user's mobile computing needs, they are still unable to provide for the full mobile computing functionality as they confine the user mobility to be within certain regions in order to benefit from services provided by fixed network access points.

Mobile ad hoc networks (MANETs) are introduced as the technology that potentially will make the nowadays illusion of mobile computing a tangible reality. MANETs are created by the mobile computing devices on an ad hoc basis, without any support or administration provided by a fixed or pre-installed communications infrastructure.

Along with their appealing autonomy and fast deployment properties, MANETs exhibit some other properties that make their realization a very challenging task. Topology dynamism and bandwidth limitations of the communication channel adversely affect the performance of routing protocols designed for MANETs, especially with the increase in the number of mobile hosts and/or mobility rates.

The Connected Dominating Set (CDS), a.k.a. virtual backbone or Spine, is proposed to facilitate routing, broadcasting, and establishing a dynamic infrastructure for distributed location databases. Minimizing the CDS produces a simpler abstracted topology of the MANET and allows for using shorter routes between any pair of hosts. Since it is NP-complete to find the minimum connected dominating set, MCDS, researchers resorted to approximation algorithms and heuristics to tackle this problem.

The literature is rich of many CDS approximation algorithms that compete in terms of CDS size, running time, and signaling overhead. It has been reported that localized CDS creation algorithms are the fastest and the lightest in terms of signaling overhead among all other techniques. Examples of these localized CDS algorithms are Wu and Li algorithm and its Stojmenovic variant, the MPR algorithm, and Alzoubi algorithm. The designers of each of these algorithms claim that their algorithm exhibits the highest degree of localization and hence incurs the lowest cost in the CDS creation phase. However, these claims are not supported by any physical or at least simulation-based evidence. Moreover, the cost of maintaining the CDS (in terms of the change in CDS size, running time, and signaling overhead), in the presence of unpredictable and frequent topology changes, is an important factor that has to be taken into account -a cost that is overlooked most of the time.

A simulation-based comparative study between the performance of these algorithms will be conducted using the ns2 network simulator. This study will focus on the total costs incurred by these algorithms in terms of CDS size, running time, and signaling overhead generated during the CDS creation and maintenance phases. Moreover, the effects of mobility rates, network size, and mobility models on the performance of each algorithm will be investigated. Conclusions regarding the pros and cons of each algorithm will be drawn, and directions for future research work will be recommended.

Acknowledgements

First, all thanks and praise are due to the Almighty Allah for giving me guidance and strength to finish this thesis.

I would like to express my thanks and deep appreciation to my supervisor Professor Otman Basir for giving me the honor of being one of his students and for providing all the assistance and guidance I needed during my research. I would like also to thank my thesis readers: Professor Eihab Abdel-Rahman and Professor Sagar Naik, for their invaluable remarks and comments.

I would like to express my deep feelings of gratitude and appreciation to my father, my mother, my grandmother, and my siblings for their unlimited support and encouragement, and to my uncles, aunts, and all my relatives for their sincere wishes.

My deepest thanks and appreciation to Dr. Ismail Elabib, Dr. Elmahdi Bouseta, Dr. Rajab Farj, Dr. Samir Albouni, Dr. Belgasim Shalouf, Dr. Ramadan Alabani, Mr. Almahdi Annajah, Mr. Mohammed Annajah, Mr. Mohammed Khoualid and Mr. Hossein Annajah for their help throughout my study.

Special thanks to my wife, Sumia, for her patience, support, and sincere prayers during the time of my research.

Thanks to my brother Abdulrahman and to my friends: Akrem Alghazal, Abduljaleel Alnetfa, Naji Alamrouni, Assadg Abdallah, Abdulmunaim Beela, Belgasim Alarabi, Atef Abdrabu, Rami Alangar, Zeyad Albasir, Imhemed Ben Yousef, Walid Elgherwi, Adel Salama, Nabil Drawil, Dr. Idris Alfgi, Ahmed Alghdamsi, Abdulmajeed Zinghina, and Abdulghafar Ashareef for their support and encouragement.

Dedication

To my parents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Due to recent advances in the computation and communication industries, portable computing devices that are capable of wireless communications are ubiquitous nowadays. Even though these devices are introduced to satisfy the user's mobile computation needs, they are still unable to provide fully mobile computing since they confine the user mobility within certain regions to benefit from services provided by fixed network access points. This limitation becomes apparent day after day as we witness increasing dependence on networking services.

To make the nowadays illusion of mobile computing a tangible reality, a networking technology that allows for instantaneous creation of temporary and reconfigurable networks has to come to existence. The multi-hop radio packet networks, also known as **mobile ad hoc networks** or **MANETs**, are prescribed by researchers as the required technology that will bridge that gap between illusion and reality. As its name implies, MANETs are created by the mobile computing devices on an ad hoc basis without any support or administration provided by any fixed or pre-installed communications infrastructure.

1.1 Advantages of MANETs

The self-control, self-organization, and ad hoc properties of MANETs are suitable for many applications in both civilian and military sectors. These applications range from providing temporary and instantly deployable communications networks in conference rooms, class rooms, and exhibitions to providing reliable communications networks in rural areas, devastated areas, and battle fields [78].

1.2 Challenges of MANETs

Along with their attractive properties, MANETs exhibit some other properties that make their realization a challenging task. Due to the mobile nature and the limited transmission ranges of MANETs' nodes, communication links between nodes are established and torn down in unpredictable fashion, as such MANETs are characterized by dynamic topologies. The self control property, a powerful feature of MANETs, puts additional burden on the nodes that they have to perform routing and other network administrative tasks cooperatively. For traditional routing protocols to discover and maintain valid routes in such dynamic topologies, a huge amount of overhead traffic is necessary; taking into account the limited bandwidth of the wireless channel used by MANETs, all traditional routing protocols are non-feasible solutions for MANETs routing.

1.3 Routing in MANETs

A routing protocol that exhibits fast and localized reactions to topology changes, keeps the control overhead generated during route search and route maintenance to a minimum, and scales satisfactorily to the increase in the rate of topology change, node density, and communications demand is eagerly sought by researchers. There have been a number of protocols proposed to address the routing problem in MANETs. These routing protocols can be classified based on the following criteria:

- Based on *which* nodes are allowed to take part in routing:
    1. **Flat routing protocols:** in this class, all nodes of the network take part in route search and packet forwarding.
    2. **Hierarchal routing protocols:** in this class, nodes are grouped into clusters that are controlled by nodes that are elected as *clusterheads*; clusterheads are connected to each other by nodes called *gateways*; the chain of clusterheads and gateways constitute a virtual backbone that can reach any node in the network. Only *clusterheads* and *gateways* are allowed to participate in route search and packet forwarding.

- Based on *When and How* routes are determined:

  1. ***Proactive* routing protocols.** In this class, every node looks for and maintains a route to each destination even if it is not currently engaged in any communication activity with that destination. The main advantage of this approach is that a route to any destination is immediately available when needed. On the other hand, the bandwidth and storage requirements are very high.
  2. ***Reactive* routing protocols.** In this class, nodes attempt to minimize the bandwidth and storage requirements by looking for and maintaining routes only when they are needed. Even though protocols of this class succeed in conserving bandwidth and storage, they exhibit some delay because of the route discovery process that is performed before any data transmission can take place.
  3. ***Hybrid* routing protocols.** In this class, nodes incorporate the advantages of the previous two classes by using proactive routing to find and maintain routes to nodes located in nearby regions and using reactive routing to find routes to nodes in distant regions.

Generally speaking, none of the reviewed protocols exhibits all the required properties; however, some of these protocols show promising performance and give good insight regarding the design methodology that should be employed in the design of the sought after optimal routing protocol.

It has been observed that flat routing protocols have poor scalability due to the massive amount of overhead generated in route search and maintenance [1, 51, 83]. On the other hand, hierarchal , a.k.a. cluster-based, routing protocols have demonstrated more scalability potential. The self-organizing control structure introduced by hierarchal protocols succeeds in localizing the reactions to topology changes by abstracting the MANET topology and presenting it as a graph of interconnected clusters. Hierarichal routing protocols incur an additional cost during the establishment and maintenace of clusters and in keeping track of node affiliation with the clusters; this additional cost is the main drawback of these protocols [1]. However, some protocols in this category use the notion of graph domination to build a connected dominating set (CDS)that functions as a virtual backbone for the MANET [6, 89, 92]. These protocols do not keep track of node affiliation and focus on node coverage [1]; hence, they eliminate the overhead associated with tracking of node af-

[1] node γ is covered if it can be reached through at least one clusterhead

filiation. Minimizing the cardinality of the CDS produces a simpler backbone and allows for shorter routes. Since it is NP-complete to find the minimum connected dominating set (MCDS) in general graphs [50], researchers resort to approximation algorithms and heuristics to tackle this problem. Among the most renowned algorithms proposed for this problem are: **Wu and Li** algorithm [92] and its variant proposed by **Stojmenovic** [85], **Alzoubi** algorithm [6], and the **MPR** algorithm [2]. These algorithms compete in terms of the size of the produced CDS, and the time spent and the overhead genenrated in establishing and maintaining the CDS in face of topology and node density changes. Even though some theoritical performance evaluation of some of these algorithms have been reported [7], to the best of our knowledge, no simulation based copmarison of these algorithms has been conducted so far. The design and the theoritical performance evaluation of these algorithms were based on the assumption that *all messages broadcasted by a given node are correctly received by its neighbors in a finite period of time*; however, this assumption is not always valid since some of the broadcasted messages get lost due to collisions resulting from the contention-based channel access schemes used in MANETs; therefor, the theoritcal performance measures may not be able to expose the actual performance and applicability potentails of these algorithms.

1.4 Thesis Organization

The main objective of this research is to investigate the ability of **Wu and Li** algorithm [92] and its variant proposed by **Stojmenovic** [85], **Alzoubi** algorithm [6], and the **MPR** algorithm [2] to function in a real environment where nodes compete to get access to the shared communication channel and where packets get lost due to collisions. Another goal of this research is to study the effects of the rate of mobility, network size, and mobility models on the CDS size, algorithm's running time, and the signaling overhead produced by each algorithm. The rest of this thesis is organized as follows:

Chapter 2 presents a comprehensive review of MANETs routing algorithms. In this chapter routing algorithms are classified into three categories based on *when and how* routs are constructed and maintained. The advantages and the disadvantages of each routing category are highlighted.

Chapter 3 presents a thorough review of MANET's clustering algorithms. These algorithms are classified based on the *graph domination* principle employed in each algorithm. The pros and cons of each class are summarized.

In chapter 4, the implementation details of the selected algorithms are presented. Chapter 5 explains the experimental set up and results. The conclusion and the future directions of this research work are presented in Chapter 6.

Chapter 2

Routing

Routing is one of MANETs' challenges that has been a topic of extensive research since MANETs were anticipated. Node mobility and the limited capacity of the shared communication channel make all routing protocols employed in wired networks inapplicable in MANETs. These routing protocols have to be modified or redesigned to be applicable in MANETs.

Inspired by the design principles of the wired networks routing protocols, researchers have proposed many routing protocols for MANETs. Generally speaking, the proposed protocols make use of distance tables, links state information, flooding, and hierarchal organization of the network to achieve routing.

The reviewed protocols can be classified into different categories based on several criteria. Presenting these protocols in a classified manner allows us to grasp the main advantages and disadvantages of the design principle common to all protocols of each category and facilitates the study of these protocols. In this chapter, the reviewed protocols are classified into ***Proactive routing***, ***Reactive routing***, and ***Hybrid routing*** depending on ***when and how*** routs are computed.

In the following sections, a summary of the most widely renowned protocols of each class are presented along with the main advantages and disadvantages of each class.

2.1 Proactive routing protocols

As mentioned earlier, in these protocols every node maintains routing information to every other node in the network even if there is no need for any of these routes at the moment, anticipating that a need for a route will arise at any time in the future. The routing information is kept in a number of different tables that are updated periodically or after topological changes. The algorithms of this group differ in the number of tables used, the type of information stored, and the mechanism of detecting topology changes and updating tables.

2.1.1 Destination-Sequenced Distance-Vector protocol (DSDV) protocol

This protocol is based on the traditional DBF, (distributed Bellman-Ford shortest path algorithm). In the DSDV protocol [79], each node keeps a table that contains an entry for every destination specifying the next node on the route to that destination and the number of hops to be traversed to reach it. Moreover, every destination's entry contains a sequence number issued by the destination itself to facilitate the detection and replacement of old routes with more recent ones. Neighboring nodes exchange their routing tables with each other periodically. After receiving routing information from its neighbors, every node updates its routing table based on the information received from neighbors, increments its sequence number and keeps its updated table ready for the next broadcast. For a given destination, the route with the most recent sequence number is always used. If two routes happen to have the same sequence number, the shorter route is used. When a node υ loses a link to a neighbor μ, υ adjusts the fields of μ's entry in the following way: it sets the number of hops to μ to infinity, and it increments the sequence number associated with μ. υ will consider μ unreachable until it receives a route to it with a finite number of hops and a larger sequence number.

In addition to the slow convergence drawback inherited from DBF protocol, the main disadvantage of the DSDV protocol is the large amount of traffic it introduces by the periodic update messages and therefore it is not able to scale to large networks.

2.1.2 The Wireless Routing Protocol (WRP)

WRP [72] is a table-based routing protocol that aims at providing the shortest route from a source to a destination. At every node, the routing information is stored in four different tables that are updated by exchanging update messages between neighbors. The names of these tables and their contents are as follows:

1. The distance table: the distance table of node υ is a matrix containing the number of hops from node υ to any other node μ through each neighbor node γ, along with the identifier of the predecessor of μ as reported by γ.

2. The routing table: the routing table of node υ is a matrix with an entry for every destination μ that stores the number of hops to μ, the predecessor on the shortest path to μ, the successor on the shortest path to μ, and a tag that indicates whether the route specified by the successor and predecessor is free of loops or not.

3. The link cost table: the link cost table of node υ provides the cost of sending information through every link connected to υ and the time elapsed since the last error-free message received over that link.

4. The message retransmission list: the retransmission list of node υ contains an entry for every update message that is still waiting for acknowledgment from some or all of the neighbors of υ. Each entry contains the sequence number of the message, an acknowledge required flag for each neighbor indicating whether that neighbor has acknowledged the update message or not yet, a retransmission counter that is decremented every time node υ sends a new update message, and a copy of the message itself. Node υ retransmits the update message if its retransmission counter reaches zero and yet some nodes have not acknowledged it. When all the neighbors acknowledge the update message that message will be deleted from the retransmission list.

The update messages exchanged between neighbor nodes carry the following information:

1. The identifier of the sending node.

2. The sequence number of the update message.

3. Update field of zero or more length that specifies the destination node, the offered distance to the destination node, and the predecessor on the new route.

4. Acknowledgment field of zero or more length that carry the sequence number of the message being acknowledged.

To insure connectivity, neighbors exchange empty update messages called *Hello* messages when there has been no data transmission between them for a specified period of time. Nodes update their routing tables either after receiving update messages from neighbors or when detecting a link failure. When node υ receives an update message from neighbor μ regarding node γ, node υ first checks whether the path offered by node μ dose not include υ as a predecessor, and whether this is the shortest path offered to node γ by all neighbors or not. If the path fulfills these two requirements, node υ updates its routing table. Node υ also updates the distance of other paths offered by neighbors other than node μ if these paths to node γ use node μ as predecessor.

The advantage of WRP is that it provides loop-free routes and it avoids temporary routing loops by using the predecessor information. On the other hand, it has significant memory requirements since it requires each node to store four routing tables; thus, WRP is unable to scale as the size of the network increases. Moreover, since nodes are exchanging *hello* messages, they are required to stay active all the time, i.e., they can not enter sleep mode to save power.

2.1.3 Global State Routing (GSR) protocol

The GSR protocol [24] is based on the same principle of the link state routing protocol, (LS), used in wired computer networks. In LS, every router maintains a map of the network topology and uses that topology map to compute the shortest routes to all destinations. Routers build their views of the network topology by getting each router to flood the link state information of all links connected to it and the identities of the routers at the other ends of those links, i.e., its neighbors. Also, routers flood any changes in link states immediately. The only improvement that GSR introduced to the traditional LS protocol is that GSR does not flood the link state packets. Instead, every node in GSR maintains a link state table based on the up-to-date information received from neighbor nodes, and periodically exchanges it with its neighbors only. Nodes attach sequence numbers to every broadcast to facilitate the usage of recent topology information. Exchanging link state tables only with neighbors reduces routing overhead; however, it slows down convergence. Moreover, as the number of nodes increases, the size of the link state tables increases; therefore, the amount of bandwidth consumed increases.

2.1.4 Fisheye State Routing (FSR) protocol

The FSR protocol [76] can be considered as an improved version of the GSR protocol; the improvement introduced by FSR is that link state table entries that

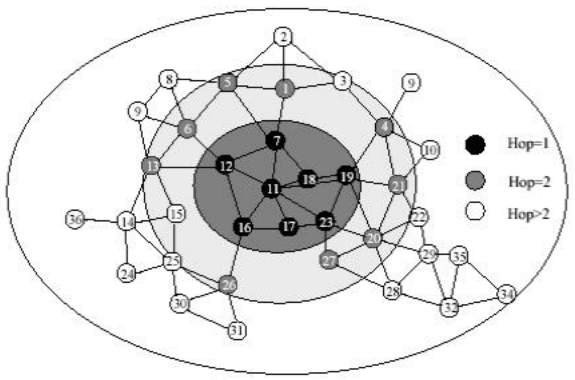

Figure 2.1: scope of fisheye

correspond to closer nodes are exchanged with neighbors, i.e., updated, more frequently than those correspond to far away nodes. This reduces the size of the exchanged link state tables and improves the utilization of the channel bandwidth. Even though less frequent exchange of link state information regarding far away nodes degrades the accuracy of routes leading to those nodes, the route accuracy increases as packets get closer to destinations. However, when the mobility of far away nodes increases, the routes to those destinations become less accurate. This difficulty can be overcome by making the exchange frequency of link state information with remote nodes proportional to the rate of mobility. Figure 2.1 illustrates the operations of FSR in an ad hoc network containing three fisheye scopes with respect to node 11. These scopes are shown as different shades of the gray.

2.1.5 Source-Tree Adaptive Routing (STAR) protocol

In the STAR [47] protocol, every node exchanges with its neighbors the prosperities of links it uses to reach other nodes. The set of links along the preferred paths used by a node to reach other nodes is called the source tree of that node. Using the source trees reported by its neighbors and the link state information of the adjacent links, every node can build partial topology graph. This graph is used to create the source tree of that node. Since the source tree contains only a subset of the links in the network, the amount of overhead information exchanged between nodes is significantly reduced. Moreover, the exchange of the routing information takes place in an event driven manner, i.e., it is not periodic. A node ϑ reports changes in its source tree to its neighbors in one of the following cases:

1. ϑ discovers a new neighbor.

2. ϑ loses all its routes to some destination.
3. The change in the cost of a path exceeds a predefined threshold.
4. The possibility of permanent loop formation.

Even though the STAR protocol reduces the consumption of channel bandwidth, it may require significant memory and processing overhead in large and highly mobile networks.

2.1.6 Distance Routing Effect Algorithm for Mobility (DREAM)

The DREAM [14] algorithm introduces new ideas to the field of routing in Ad hoc networks. As in all proactive protocols, in DREAM every node collects routing information about other nodes even if there is no need for it; and it uses this information to trigger a directed flood to deliver messages to the intended destinations. The direction of the flood is calculated in an on-demand basis using the stored information, i.e., this algorithm exhibits some reactive features. The novelty of this algorithm is in the type of information it collects and how often this information is updated. Every node utilizes a GPS system to determine its location information and it disseminate this information to all other nodes that store it in location tables. Location information can be sent in smaller packets than those used to carry distance information; therefore, DREAM is more efficient in terms of bandwidth utilization. The designers of DREAM have noticed that location information of nodes that are far away from each other seems to change in a rate that is slower than the actual one. Thus, they make the rate of location-information update proportional to the distance between nodes. That is every node sends location-information updates to nearby nodes more frequently. This is accomplished by including lifetime information in the update message. Lifetime, or time to live (TTL), is expressed in terms of the number of traversed hops. Messages with short TTL are sent more frequently. Moreover, the frequency of location-information update is proportional to nodes mobility, i.e., highly mobile nodes disseminate their location information more frequently. Based on the location-information of the destination, the source node can determine the direction of the flood process used to deliver messages to that destination.

2.1.7 Multimedia support in Mobile Wireless Networks (MMWN) protocol

The MMWN protocol [62] is a hierarchal link-state routing protocol that takes the QoS requirements into consideration in the process of route establishment. In the MMWN protocol, nodes are classified into endpoints and switches. Endpoints organize themselves in cells, called level-0 clusters, around switches called cell-heads. Every endpoint belongs to only one cell-head switch and it is in direct contact with that switch. Switches organize themselves in clusters and they elect one of them to become the cluster leader; cluster leaders organize themselves in higher-level clusters and elect leaders for these clusters. The number of members allowed for each cluster controls the number of hierarchal levels. Figure 2.2 explains the hierarchal structure of the network according to the MMWN protocol. Every level-n cluster, $n > 0$, is a parent cluster of some level-(n-1) clusters and grandparent of some level-(n-2) clusters; for example in Figure 2.2 cluster B is the parent of clusters E and F while cluster U is their grandparent.

Switches are classified into border switches, i.e., switches that are in contact with endpoints from other clusters that belong to the same hierarchal level, and interior switches that include all other switches of the cluster; for example, in Figure 2.2, switches X and S are border switches of cluster E; and all other switches of that cluster are interior ones. Neighbor clusters of the same level communicate through virtual gateways, VGs, which are pairs of peer border switches that are in contact with each other. A virtual link, VL, connects two non-neighboring clusters of the same level through the virtual gateways of a third cluster that is a common neighbor to both of them; in Figure 2.2, switches X and Z are border switches that form a virtual gateway between clusters E and D, and the link between switches Y, X and Z is a virtual link that connects clusters F and D. Every endpoint and every switch has a globally unique identifier known as the endpoint-id and the switch-id, respectively. Moreover, every level-n cluster, $n > 0$, has a unique ID among its siblings. The ID of any level-0 cluster is the ID of its cell-head switch. The address of any level-n cluster is its local ID prefixed by the IDs of all its parents; for example the address of cluster E in Figure 2.2 is U.B.E, and the address of any endpoint is the same as that of its cell-head switch, i.e., its level-0 cluster. Moreover, every cluster has a switch known as the location manager, LM, and it is the only member of the cluster that keeps track of the locations of other endpoints. Also, every cluster has a QoS manager that is responsible of generating the link state information of all links within the cluster and between clusters of the same level. In level-0 clusters,

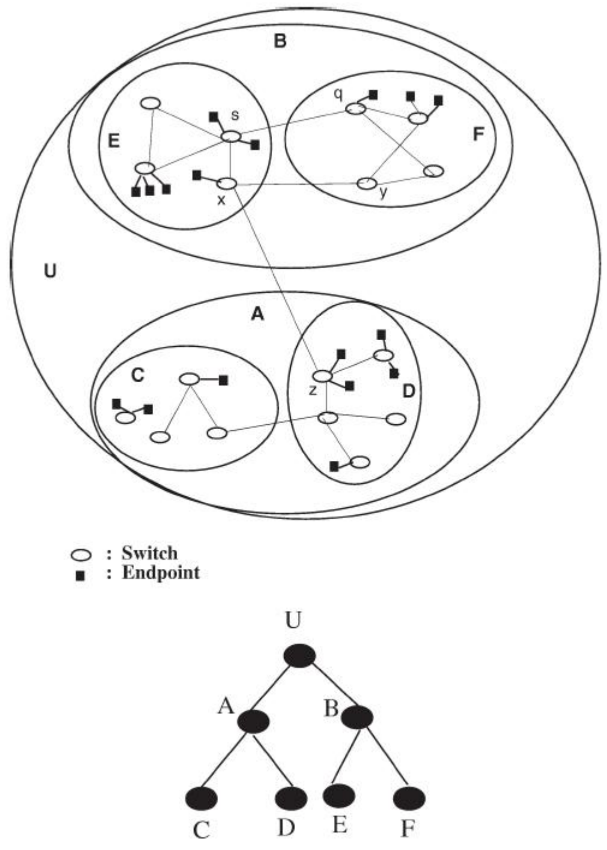

Figure 2.2: The hierarchal structure of MMWN

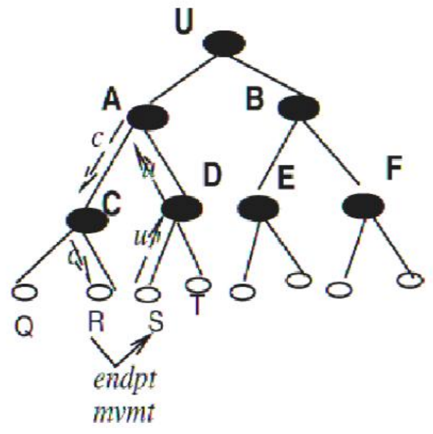

Figure 2.3: The location table update due to node movement

the switch acts as the cluster's QoS manager. The QoS characteristics of the cluster links are included in its link state information. The leader of level-n cluster sends the link state information to the leaders of its sibling clusters and the leaders of its children clusters. Based on the received link state information, the leaders of level-0 clusters can build a topology map of the network and can select routes based on the QoS requirements. Whenever an endpoint wants to start a session with other endpoint, it provides the destination ID to its cell-head switch that consults its location manager about the location of the destination and then determines the route that satisfies the QoS requirements. This algorithm tries to mimic the cellular phone system by selecting nodes that has low relative mobility to function as switches for nodes in their proximity and as a result limits the number of nodes needed to keep track of link state and location information and hence reduces the routing overhead. The main disadvantage of this algorithm is that whenever an endpoint moves from a cluster to an other, all the location managers of the clusters within which the movement occurred have to be updated and this might lead to inconsistency in the location tables due to propagation delays. Figure 2.3 shows an example of node movement and the location tables' updates that have to be done.

2.1.8 Cluster-head Gateway Switch Routing (CGSR) protocol

The CGSR protocol [30] is another hierarchal routing protocol in which nodes are grouped into clusters. Each cluster has a cluster-head elected from its members. Nodes that are in the communication range of more than one cluster-head are called gateways. The CGSR protocol differs from the MMWN protocol in that:

1. There is only one level of clustering.
2. Cluster-heads use distance information to select routes (in fact it uses the DSDV to route messages among cluster-heads through gateways).

In the CGSR protocol, every cluster-head maintains two tables; one is used to store the affiliation information and it has an entry for each node that stores the node ID along with the ID of the cluster-head of that node stamped with a sequence number to facilitate the identification of stale affiliation information. Cluster-heads exchange these affiliation tables periodically through gateways. The other table, (also maintained by gateways), is used to store routing information and it has an entry for each cluster-head stamped with a sequence number generated by

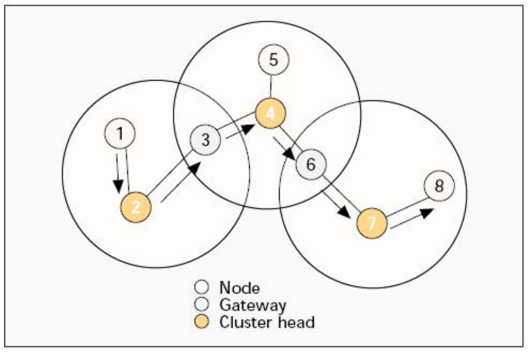

Figure 2.4: The CGSR hierarchal structure and routing mechanism

the cluster-head itself along with the ID of the gateway (cluster-head in the tables maintained by gateways), on the shortest bath to that cluster-head. Cluster-heads and gateways exchange their routing tables periodically. Whenever a node has a message to send to another node, it first sends it to its cluster-head that finds the destination's cluster-head by looking into its affiliation table; the cluster-head of the source node then uses its routing table to identify the gateway that provides the shortest path. This gateway uses its routing table to identify the next cluster-head on the shortest path and so on. Figure 2.4 depicts the hierarchal structure imposed by CGSR and the mechanism of message delivery.

This algorithm tries to reduce the routing overhead by allowing only cluster-heads and gateways to maintain and exchange routing information. The major disadvantage of the CGSR protocol is that cluster heads and gateways are heavily loaded so that they consume more power than other nodes; moreover, in highly mobile networks, there might be significant overhead associated with maintaining clusters.

2.1.9 Hierarchal State Routing (HSR) protocol

The HSR protocol [77] is a hierarchal link state routing protocol that resembles to a great extent the MMWN protocol, except in the location management aspect which is considered a drawback of the MMWN protocol. In the HSR protocol nodes are grouped in clusters and the nodes of every cluster select a cluster head; the cluster heads are grouped in higher level clusters that have cluster heads. There are three types of nodes in every level-0 cluster, namely: cluster head nodes, gateway nodes, and internal nodes; for example, in Figure 2.5, nodes 1, 4, and 6 are cluster heads; nodes 3 and 7 are internal nodes; while nodes 2 and 5 are gateways. The cluster

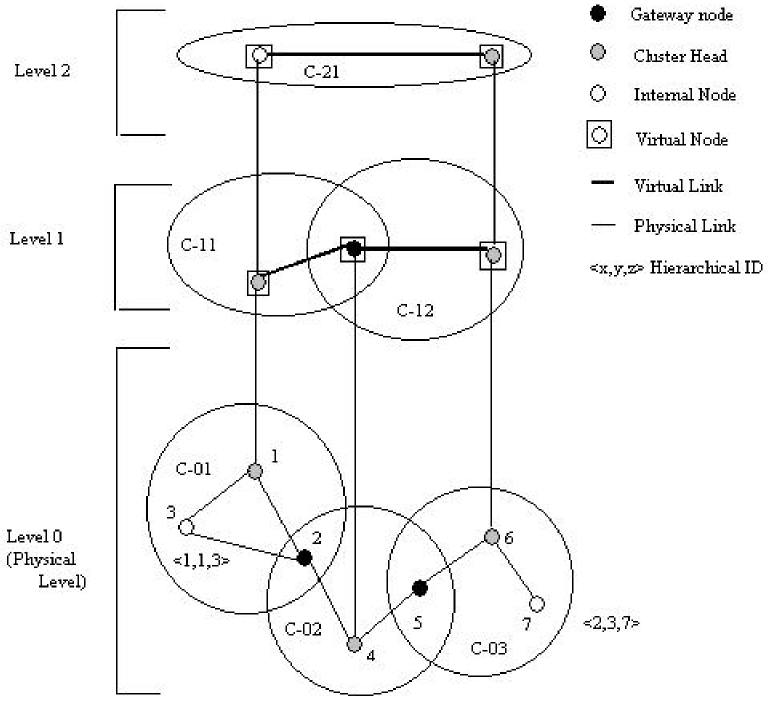

Figure 2.5: HSR hierarchal structure and addressing mechanism

heads of higher level clusters are called virtual nodes and the links between these nodes are called virtual links.

In addition to its unique ID (its MAC address), every node has a hierarchal physical address known as the HID address and it is composed of the node ID prefixed by the IDs of all its parent clusters; for example, in Figure 2.5, the HID of node 3 is $< 1, 1, 3 >$. Nodes of the same cluster, at all levels, exchange link state information about links belonging to the same level; moreover, they exchange a summary of the link state information of the lower levels. Higher level cluster heads flood down the information of the network connectivity at their levels to the nodes of the lower levels so that level-0 nodes can have a prospective about the network topology. Using the HID address, a packet can be delivered to any destination from any source; for example, to send a packet form node 3 whose HID is $< 1, 1, 3 >$ to node 7 whose HID is $< 6, 6, 7 >$, the packet first is delivered to node 1 which is the cluster head of node 3; using the HID address of the destination, node 1 realizes that its job is to deliver the massage to node 6 which is the cluster head of node 7. Using the level-1 link state information, node 1 finds a logical link to node 6; this link passes through nodes 2-4-5. HSR provides a remedy to the problem of location management of mobile nodes by introducing the concept of logical subnet which is a group of nodes that have common characteristics (e.g., professionals belonging to the same company, students of the same class, etc.), who are not necessarily close to each other in terms of the physical distance. In addition to the HID address every node has a logical address in the form of $< subnet - host >$. Each subnet has a home agent that is responsible of keeping track of the HIDs of its members. Home agents advertise their HIDs to the higher level cluster heads, so that cluster heads know the HID of every home agent at all times. Every node updates its home agent about its HID either periodically or on event-driven basis. When a node moves to a new cluster, only its home agent is updated, so that the communication overhead is reduced and the inconsistency problem that arises from updating many agents is avoided. Whenever a node wants to communicate with any other node, first it extracts the subnet information from the destination's logical address and consults it cluster head to get the HID of the destination's home agent; it then uses this HID to query the home agent about the destination's HID. Once the source knows the HID of the destination, the communication takes place without involving the home agents. HSR reduces the routing and control overhead significantly. However, as for all hierarchal protocols, it involves overhead related to clusters formation and maintenance.

2.1.10 Optimized Link State Routing (OLSR) protocol

The OLSR protocol [56] is an optimized version of the traditional link state algorithm. This protocol minimizes the size of the update message by allowing nodes to include the link state information of only a subset of their neighbors, which are called the node's multi point relay selectors. Moreover, it introduces a mechanism to control the flooding process by allowing only a subset of node's neighbors to retransmit its update massages; these nodes are called the multi point relays (MPR). Neighboring nodes, other than the MPRs, receive and process the link state packets; however, they are not allowed to retransmit them. Every node constructs its MPR set by periodically exchanging its one hop neighborhood information with its neighbors. The exchange of such information allows every node to discover its two hop neighborhood and to find the minimum number of one hop neighbors that allows it to reach any node in its two hop neighborhood; for example, in Figure 2.6, the MPR set of node A consists of nodes B, C, K, and N. In this example node A is called the MPR selector of nodes B, C, K, and N.

Every node sends a *Hello* message to its neighbors; *Hello* messages contain the IDs of its MPRs; so that the MPRs become aware of their MPR selectors and hence retransmit any link state packets received form these selectors. The OLSR

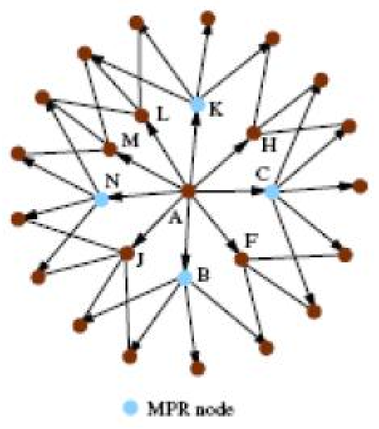

Figure 2.6: The MPR set

protocol depends on the MPRs to deliver messages between any source-destination pairs. To achieve this mechanism every MPR node informs its neighbors about its MPR selectors; MPRs of the announcer node retransmit the received information to their MPRs until the entire network is covered. Network nodes use this information to build topology tables that represent the topology map of the network. These topology tables are used to compute routing tables. In addition to the routing tables and topology tables, every node N in the OLSR protocol maintains two other tables, namely:

1. Neighbors Table: this table contains an entry for each neighbor wherein it stores the neighbor ID, the status of the link to that neighbor: (uni-directional, bi-directional, or MPR which is the case if that neighbor is a MPR node).

2. MPR selectors Table: this table contains an entry for every neighbor that has selected node N as a MPR node.

2.1.11 Topology Broadcast Reverse Path Forwarding (TBRPF) protocol

The TBRPF protocol [18] is a link state routing protocol in which a reduction in the routing overhead is achieved by making every node broadcasts differential link state information, i.e., only changes that happened after the last transmission are reported. Moreover, nodes send only parts of their source trees to their neighbors;

these parts include only links to nodes that the sender anticipates one of its neighbors may use as part of its shortest path to other nodes. Furthermore, this protocol uses the reverse path forwarding mechanism (RPF) to limit the flood process. RPF allows node ϑ to retransmit packets from source node μ if and only if these packets arrived on the link that ϑ would use as part of its shortest path to μ.

2.1.12 Characteristics of the proactive routing protocols

- In general, flat routing algorithms are not scalable due to the large routing over head they introduce to the network.
- The OLSR and DREAM protocols are the most scalable flat routing algorithms.
- Hierarchal routing algorithms are more scalable than the flat ones. However, in highly mobile networks they introduce extra processing overhead related to cluster formation and maintenance.

2.2 Reactive routing protocols

As mentioned earlier, reactive protocols, also known as on-demand protocols, look for and maintain routes only when they are needed. The on-demand nature of these algorithms frees nodes from keeping track of changes in link states and distances when there is no communication activity taking place. This approach results in significant optimization of the channel use; however, it results in some delay due to the route discovery stage.

2.2.1 Dynamic Source Routing (DSR) protocol

The DSR protocol [61] is considered as an ideal example of on-demand routing protocols. In this protocol, every node maintains a route cache wherein it caches routes for destinations with which it has communicated recently or it has learned of by listening to the communication activities tacking place around it. Whenever a node has data packet to send to any other node, it checks its cache to see if it has a fresh enough route for that destination. If so, it embeds that route in every data packet to be sent. Otherwise, the source node starts a route discovery process by creating query packet that contains the destination address, the source address, and a unique sequence number and floods it to the entire network. Upon the reception of the query packet, any intermediate node that does not have a cached route for the destination attaches its addresses to the query packet and retransmits it. Intermediate nodes retransmit the query packet if it was not seen before; otherwise, they ignore it. This process continues until the query packet reaches the destination node or an intermediate node that has a route to the destination. When the destination node receives the query packet, it sends a route reply packet to the source node; the route reply packet uses the route embedded in the query packet to reach the source packet or it uses another route if the communication links are not bidirectional (this route can be found in the route cache or by another route discovery process). On the other hand, if the reply is generated by an intermediate node, that node concatenates its cached route to the destination to the addresses of the nodes traversed by the query packet during the search process. The disadvantage of the DSR protocol is that since data packets have to carry the entire routing information, the size of data packets increases with the increase of the network diameter

2.2.2 Ad hoc On-demand Distance Vector (AODV) protocol

The AODV protocol [38] can be thought of as a combination of the DSDV and DSR protocols. It takes the sequence numbers and the periodic *Hello* messages of the DSDV protocol and the route discovery mechanism of the DSR protocol. The difference between the AODV and DSR protocols is that in the AODV protocol data packets carry only the address of the destination; every node on the path knows the ID of the next node on the route to the destination. In the AODV protocol every node maintains two monotonically increasing numbers :

1. The *broadcast_id* that is incremented every time the node initiates a route discovery process.

2. The *node_sequence_no* that indicates freshness of the routing information related to a particular node.

Whenever a node wants to send a message to any other node for which it has no valid route, it starts a route discovery process by creating RREQ packet that carries: the source's ID, the source's *broadcast_id*, the destination's ID, the latest sequence number of the destination that the source has heard of, and a hop count; then it floods this RREQ to the network. Any intermediate node, γ, that satisfies the following two conditions:

1. It does not have a fresh enough route to the destination, i.e., route with a sequence number that is larger than or equal to that indicated in the RREQ packet.

2. It has not seen this RREQ packet before.

proceeds as follows:

1. γ increments the hop count field of the RREQ packet and retransmit it to all its neighbors.

2. γ stores the address of the node from witch it received the RREQ packet to use it later to forward the route reply packet to the source node.

When the RREQ packet reaches the destination node or an intermediate node that has fresh enough route to the destination, a route reply packet, RREP, is generated and sent back to the source node. During the propagation of the RREP packet toward the source node, every node that receives a copy of the RREP packet sets an entry in its routing table that contains the source's ID, the destination's ID, the destination's *Sequence No*, the ID of the next hope node toward the destination, and the number of hops traversed so far. Nodes ignore any other copies of the RREP packet unless they contain a higher sequence number for the destination or provide routes with less hop count. When the RREP packets reach the source, the route traversing the smallest number of hops is selected. Even though the AODV protocol can adapt to highly dynamic networks, nodes may experience long delays during the route discovery phase.

2.2.3 Routing On-demand Acyclic Multi-path (ROAM) protocol

The ROAM protocol [82] is a reactive routing protocol that provides loop-free multiple paths to destinations using internodal coordination operation called diffusing computations. The main problem that ROAM solves is the searching-to-infinity problem in which a source node continues to search for a given destination by flooding the network with query packets even if that destination is unreachable. The solution introduced by ROAM is the use of routing tables that mark a destination as unreachable if there is no route that leads to it. Thus, the node will not participate in any search process for that destination in the near future. Moreover, ROAM allows all nodes that participate in the search process to benefit from it by either getting a finite distance route to the destination or by marking the destination as unreachable. ROAM is not pure on-demand protocol because nodes use distance tables and routing tables that might contain entries for routes that are not used at the moment. The flood process is reduced by storing multiple paths to the same destination that are ranked by their distance. Routes that are shorter than a given length, known as the feasible distance (the distance of previous shortest path), are kept for future use if the shortest one (the one stored in the routing table) fails or its cost, i.e., length, increases. If there is no alternative route or if all alternatives become invalid, the source uses the flood process to find other set of routes. The result of this flood process is made common to all nodes participated in the flood. Another novelty of ROAM is that, every time a node detects a change in the cost of one of its links that exceeds a predefined threshold, that node informs its neighbors so that they update their routing tables based on the new changes. Although ROAM achieves better network connectivity, in highly mobile networks where changes in link costs are frequent, it may prevent nodes from entering sleep mode and hence it increases the power consumption.

2.2.4 Light-weight Mobile Routing (LMR) protocol

The LMR protocol [34] is one of the on-demand routing protocols that use the link reversal approach to establish a directed acyclic graph, DAG, rooted at the destination. This graph offers loop-free multiple paths to the destination. Nodes maintain routes only to their neighbors; whenever a node wants to send data to a non-neighbor node, it creates a query packet that contains a sequence number, the source ID, the destination ID, and the ID of the transmitting node. Intermediate nodes keep track of query packets they have seen and retransmit any query one time only. Also they keep the IDs of the transmitting node and the destination node. When the query reaches the destination node or an intermediate node that has a route to the destination, a reply packet that includes the source ID is generated and sent back to the node from which the first copy of the query was received. This reply packet propagates through the network until it reaches the source node. The directions of the links over which the reply propagated are directed in the opposite direction of the reply propagation. Links that are directed into a node are called upstream links, and those directed out of a node are called downstream links. LMR offers multiple paths to the same destination so that when a path fails, there will be other paths to replace it. Figure 2.7 shows an example of LMR route construction. LMR reduces the routing overhead through limiting and localizing the effects of link failure by prohibiting nodes from announcing link failures unless a node loses all its downstream links; in this case, this node informs all its upstream neighbors using failure query packets (FQ) that contains the identifiers of the generating node and the destination node. Upon the reception of the FQ packets, the upstream nodes erase the routes that go through the failed link; any upstream node that has an alternative route to the destination informs the FQ generating node using reply message; otherwise, it propagates the FQ packet to its upstream neighbors. Figure 2.8 shows an example of route maintenance in LMR.

The drawback of LMR is that it can result in temporary invalid routes in case of network partition. Even though it has been shown that these invalid routes will eventually be erased, there is no finite bound for the time required to get rid of these invalid routes.

2.2.5 Temporally Ordered Routing Algorithm (TORA)

The TORA algorithm [35] is a source initiated routing algorithm based on the link reversal protocol proposed for wireless mobile networks; TORA aims at providing loop-free multiple paths between any source/destination pair and it is considered to be a descendant of the LMR protocol due to the fact that TORA uses the same directed acyclic graph (DAG) creation procedure used by LMR. However, the advantage of TORA over its ancestors ([34, 45]) is that TORA can detect a network partitioning and erase all routes to the unreachable destinations in a finite time. By localizing the effect of link failure to the neighborhood in which the failure took place TORA reduces the routing overhead in highly mobile ad hoc networks; however, this may lead to the use of non-optimal routes after the failure of the optimal ones. The main idea of TORA can be explained by the flow of water in a network of pipes toward a sink, i.e., a destination. In this analogy, pipes represent links and junctions between pipes represent nodes. Every node, except the destination, has a relative height with respect to a reference level defined by some neighbor. The definition of the reference level contains information about the time at which the reference level was defined, and that is the reason behind the name Temporally Ordered. The height metric of a node, γ, is expressed as

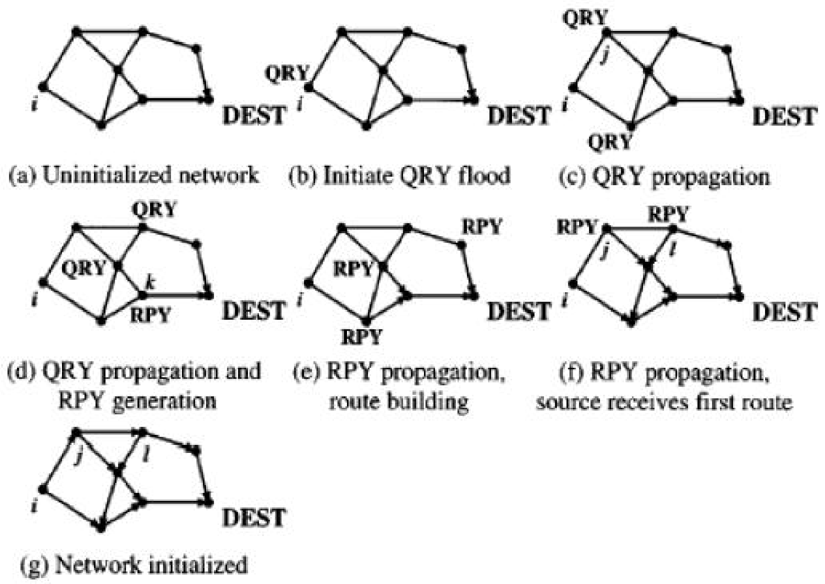
QRY
QRY
QRY
RPY
RPY
RPY
i
j
k
l
DEST
(a) Uninitialized network
(b) Initiate QRY flood
(c) QRY propagation
(d) QRY propagation and RPY generation
(e) RPY propagation, route building
(f) RPY propagation, source receives first route
(g) Network initialized

Figure 2.7: LMR route construction

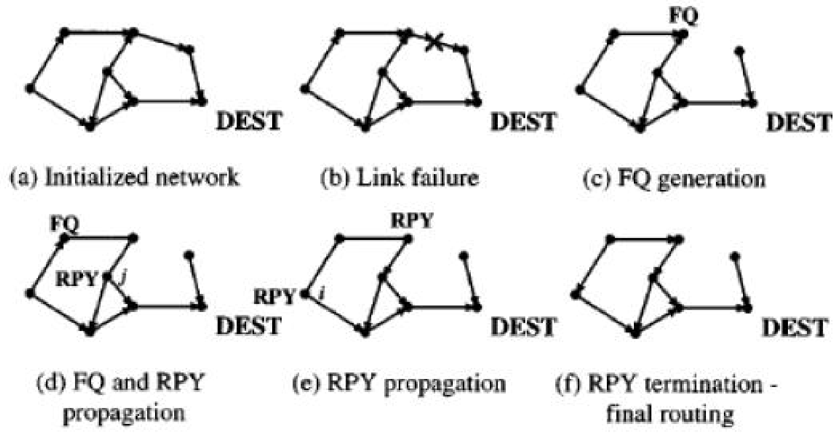

Figure 2.8: LMR route reconstruction

a quintuple comprising two parts: a reference level (the first three components), and the offset from that level (the last two components). The components of the quintuple are:

1. The logical time of the link failure,
2. The ID of the node defining the reference level,
3. A reflection indicator bit.
4. The offset from the reference level,
5. The ID of node γ.

Heights are assigned in lexicographical order, i.e., node $i = (i_1, i_2, .., i_5)$ is considered higher than node $j = (j_1, j_2, .., j_5)$ if and only if $i_1 > j_1$, or $i_1 = j_1$ and $i_2 > j_2$, etc. Links are directed from higher nodes toward lower nodes. The destination , ϑ, is always assigned the lowest height, namely its height is always represented by the following quintuple (0, 0, 0, 0, ϑ's ID). Route discovery is accomplished using the same query-reply mechanism used in LMR with the exception that query packets always reach the immediate neighbors of the destination. Whenever a node has a data to send to some destination, it creates a query packet containing the destination ID and floods it to the network. Usually, the route discovery process results in the creation of multiple loop-free paths; Figure 2.9 shows an example of the route discovery process. Whenever the last outgoing links of a given node fails, that node starts a maintenance process by defining a new reference such that its new height will be greater than the heights of all its neighbors; moreover, it informs its neighbors about its new height so that they adjust the directions of their links according to the current heights. The definition of new reference level may lead to the reversal of one or more links to become outgoing links and hence it creates new route to the destination; however, link reversal does not create new routes in all cases. If the reversal process due to link failure causes any neighbor to lose all of its outgoing links, that neighbor reacts by creating another reference level that has the reflection indicator bit set to 1 to distinguish between references resulted from link failure and those resulted from link reversal (known as *reflected levels*). If all neighbors of a given node, ϑ, have height quintuples with reflection indicator set to 1, the loss of all outgoing links of ϑ leads it to conclude that the network is partitioned and hence the destination is unreachable. ϑ informs its neighbors about the network partitioning so that they erase all invalid route. The fast discovery of network partitioning is the main advantage of TORA over LMR. Figures 2.10 and 2.11 show an example of TORA route maintenance while Figure 2.12 shows the erasing of invalid routes due to network partitioning. Although routes ultimately converge, they may suffer from temporary oscillations before convergence.

2.2.6 Associativity-Based Routing (ABR) protocol

The ABR protocol [87] is a reactive routing protocol in which routes are established using the same Query-Reply approach as that used in DSR [61]. The novelty of the ABR protocol is that routes are selected based on the temporal stability of their links. In the ABR protocol, nodes use *Hello* messages to discover their neighborhood. Every node keeps a list of neighbors wherein it stores the associativity degree of each neighbor based on the number of consecutive *Hello* messages received from it. The higher the number of consecutive *Hello* messages received over a link the better is the temporal stability of that link. Whenever the connectivity with a neighbor is lost, the associativity degree with that neighbor is reset. When the first copy of the RREQ packet is received by an intermediate node, that node adds its ID and the connectivity degree of the link over which the packet was received to the RREQ packet and transmits it to its neighbors. A route reply packet (RREP) is generated by the destination node or any intermediate node that has a valid route to the destination. A RREP is not generated immediately after the reception of the first RREQ; instead, the node generating the RREP waits for a while anticipating that other RREQs will arrive over more stable links. The sum of the associativity degrees of the links of a given route represents the overall stability of that route. Routes are selected based on their overall stability; if two routes have the same overall stability, the shorter one is selected. The ABR protocol tries to minimize the overhead of route recovery by localizing the recovery process to the neighborhood of the failed link. Whenever a link failure happens, the node (γ) immediately upstream of the failed link generates a short living, localized query packet (LQ) and sends it to its neighbors, the time-to-live of the LQ is set to one or two hops. If there is no replay after LQ-timeout period, a notification is sent to the upstream neighbor of γ that will try to repair the failure by sending another LQ packet. If the upstream nodes up to the middle of the path were not able to repair the route, a notification message is sent to the source node to initiate another route discovery process. Although routes offered by the ABR protocol are not necessarily the shortest ones available, they tend to last longer. The main disadvantages of ABR is that it forces all nodes to stay active all the time discovering their neighborhood ;

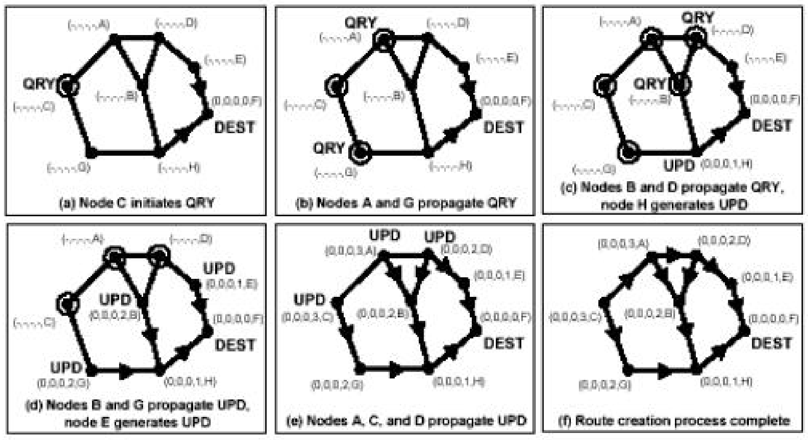
(a) Node C initiates QRY
(b) Nodes A and G propagate QRY
(c) Nodes B and D propagate QRY, node H generates UPD
(d) Nodes B and G propagate UPD, node E generates UPD
(e) Nodes A, C, and D propagate UPD
(f) Route creation process complete
QRY
UPD
DEST

Figure 2.9: TORA route construction

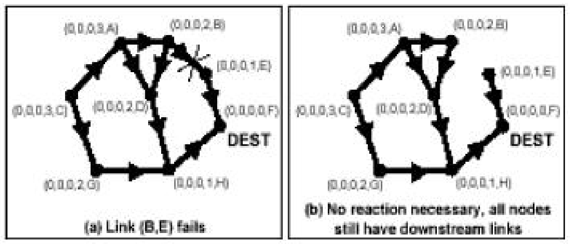
(a) Link (B,E) fails
(b) No reaction necessary, all nodes still have downstream links
DEST

Figure 2.10: Failure that triggers no repair

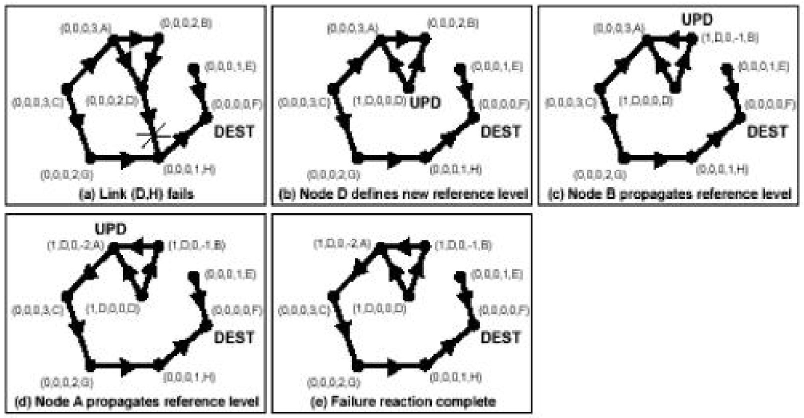
(a) Link (D,H) fails
(b) Node D defines new reference level
(c) Node B propagates reference level
(d) Node A propagates reference level
(e) Failure reaction complete
UPD
DEST

Figure 2.11: Repairing a failed route

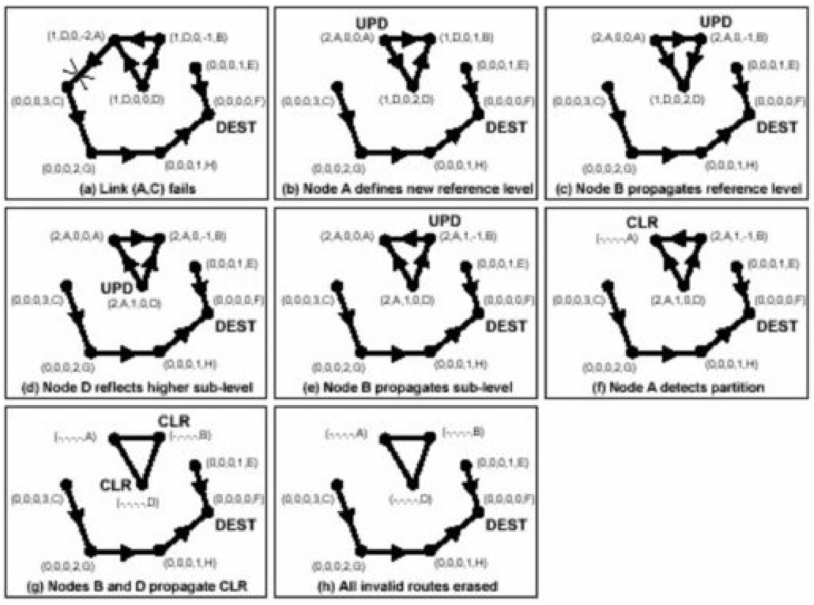
(a) Link (A,C) fails
(b) Node A defines new reference level
(c) Node B propagates reference level
(d) Node D reflects higher sub-level
(e) Node B propagates sub-level
(f) Node A detects partition
(g) Nodes B and D propagate CLR
(h) All invalid routes erased
UPD
CLR
DEST

Figure 2.12: Network partitioning discovery and invalid route erasing.

hence, it may result in high power consumption. Moreover, the ABR protocol does not maintain multiple routes to the same destination, i.e., a route repair is needed every time a route failure occurs.

2.2.7 Signal Stability Adaptive routing (SSA) protocol

The SSA protocol [40] can be considered as a descendant of the ABR protocol because it employs the same principle of selecting routes by destinations based on route stability. Using stable routes, that are not necessarily the shortest ones, reduces the rate of route failure and hence improves the efficiency of bandwidth use. The SSA protocol differs from the ABR protocol in that the SSA protocol uses more sophisticated stability measure based on signal strength and location stability. Neighbors periodically exchange *Hello* messages with each other. Every node maintains two tables:

1. The signal stability table (SST) in which the average signal strength of *Hello* messages received from each neighbor is stored.
2. The routing table (RT) in which the next hop on the path to every destination is stored.

At anytime, the number of consecutive *Hello* messages received with certain signal strength from each neighbor is used as a measure of the location stability of that neighbor. Once the location stability of a given neighbor exceeds a certain threshold, that neighbor is considered a strongly connected neighbor and an entry for it is added to the routing table. The reception of a *Hello* message of weak signal strength from a given neighbor resets its location stability to zero, marks it a weakly connected neighbor, and deletes it from the routing table. When a node needs a route to a particular destination, it sends a RREQ packet to its neighbors. An intermediate node that receives a RREQ packet from a strongly connected neighbor retransmits the RREQ packet to its neighbors. Only the destination is allowed to generate route reply. Preference is given to strongly connected routes for their tendency to live longer. If there is no strongly connected chain of links between the source and the destination, no RREP will be generated; hence, the RREQ will time out and the source will initiate another RREQ with relaxed signal strength requirements. A route failure triggers the transmission of an error message to the source node to erase the invalid route and initiate new route discovery process. Although the SSA protocol produces longer living routes, it introduces long delay due to the fact that only destinations are allowed to create route replies. Moreover, allowing only source nodes to repair routs incurs more delay by spending extra time in informing source nodes about route failures.

2.2.8 Relative Distance Micro-discovery Ad hoc Routing (RDMAR) protocol

The RDMAR protocol [3] is a loop-free routing protocol especially designed for large ad hoc networks that exhibit moderate rate of topological changes. RDMAR tries to minimize the routing overhead by localizing the query flooding used for route discovery and by localizing the reactions to link failures to small regions of the network geographical span. This is achieved by exploiting the information of the relative distance between the source and the destination expressed in terms of number of hops. Every node maintains a routing table with an entry for every reachable destination; this entry stores the destination ID, the nest hop on the path to the destination, the relative distance to the destination in terms of number of hops, the time of the last update of the relative distance, and the remaining time before route expiration. Whenever a node, γ, wants to send a packet to another node, μ, for which it has no valid route, it sends a RREQ packet to all its neighbors asking for a route to μ. If there was no previous communications between γ and μ, the RREQ flood will cover the entire network; however, if γ has communicated with μ previously, γ uses its information about the previous relative distance and the time elapsed since the last update of the relative distance to compute a new relative distance. The new relative distance is used as the time-to-live (TTL) of the RREQ. In the computation of the new relative distance, an average speed and average transmission range are used. Upon the reception of the RREQ by an intermediate node, this node sets the next hop node on the route to the source to be the node from which the first RREQ copy was received; and then it retransmits the RREQ to its neighbors. All subsequent copies of the same RREQ are discarded. In order to avoid using stale information provided by intermediate nodes that might lead to routing loops, only the destination node is allowed to generate route reply packet (RREP). Whenever an intermediate node receives a RREP, it compares the hop count of the offered route with that of the route stored in its routing table (if any); the route with the smallest hop count is stored in the routing table. If all attempts made by an intermediate node, ϑ, to forward a packet fail due to link failure, this node proceeds as follows:

- If the failure is closer to the destination node, ϑ initiates a route discovery process to find a new route.

- If the failure is closer to the source node, ϑ sends a failure notification packet (NF) to neighbors that use it to reach the destination whose route has failed. These neighbors are stored in a list called the *dependent list*. Upon the reception of the NF packet, every node in the *dependent list* removes the entry associated with that destination from its routing table; moreover, if it has no alternative route to the destination, it forwards a copy of the NF packet to all members of its own *dependent list*. To guarantee the delivery of the NF packet, any node that receives the NF packet uses its cached route to the source to deliver this error message.

Although the RDMAR protocol conserves a significant portion of the channel bandwidth without using any special equipments like GPS, when there is no previous communication between the end points, it acts in a pure flooding fashion.

2.2.9 Location-Aided Routing (LAR) protocol

The LAR protocol [63] is an on demand routing protocol that exploits the location information of the nodes to confine the flood process to a limited region of the network geographical span and hence reduces the routing overhead. While the RDMAR protocol specifies the region of the search process in terms of the number

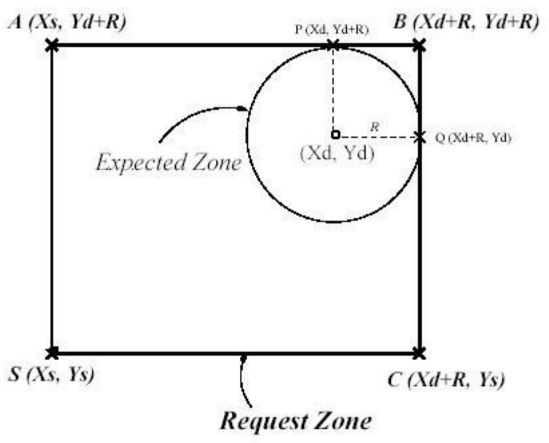

Figure 2.13: The expected zone and the request zone concept

of hops separating the session's end nodes, the LAR protocol uses the location coordinates of the end nodes provided by GPS system to achieve the same goal. In the LAR protocol, if the source node knows the location information, l, of the destination node at time, t_0; then it can estimate the location of the destination node at any later time t_1. Because the source uses the average speed, v, of the destination; the location estimate is a zone called the **expected zone**. In general, the expected zone is a circle with radius equal to $v*(t_1-t_0)$ centered at l. The more information available about node movement, the narrower is the expected zone; for example, if the direction of movement of the destination is available, the expected zone will be of a semi-circular shape. Another term introduced by the designers of the LAR protocol is the **request zone**, which is the smallest rectangular region that contains the source node and the entire **expected zone**. Figure 2.13 explains the concepts of expected zone and request zone.

Only nodes located in the **request zone** can participate in the flooding process. For a given route search, every node must be able to know whether it is inside or outside of the **request zone** in order to decide whether to take part in the route search or not. Two different schemes are used to specify the request zone:

- **The first scheme:** in this scheme the source node embeds the coordinates of the four corners of the **request zone** in the RREQ packet, so that intermediate nodes are able to determine whether they are inside or outside this zone.

- **The second scheme:** in this scheme the source node embeds in the RREQ packet the destination coordinates, which might be out dated, and its distance to this destination. Every intermediate node compares its own distance to the destination with that embedded in the RREQ packet; if its own distance to the destination is less than that embedded in the RREQ packet, the intermediate node modifies the RREQ packet by replacing the old distance with its own distance and retransmits the RREQ packet to its neighbors.

In both schemes, only the destination is allowed to generate the RREP packets. The set up of forward routes (during the RREQ propagation) and backward routes (during the RREP propagation) is done the same way as in the AODV protocol [38]. Although the LAR protocol reduces the routing overhead and, in most cases, discovers the shortest routes to destinations, it assumes the availability of GPS equipped nodes which is not always a valid assumption. Moreover, if the location information is not available; for example, in the first communication session, or when that information is not valid due to high rates of mobility or to long time periods of no communication, wide request zones that may cover the whole network are used, i.e., pure flooding is employed.

2.2.10 Ant colony based Routing Algorithm (ARA)

The ARA algorithm [52] is based on the ant colony meta heuristic technique, which is a multi-agent system of simple agents called ants. The individual ants; i.e., agents, are so simple that they can only achieve simple tasks; however, their collaboration results in a very powerful problem solving tool. The most fascinating property of this technique is that no direct communications take place between the agents during the solution development phase; agents communicate indirectly by modifying some conditions of the local environment; this kind of communication is called stigmergy. Ants modify the environment, and hence communicate, by laying a substance called pheromone on the routes they use from their nest to the food source. It was found that ants select routes based on the level of pheromone concentration on these routes. Routes that have higher pheromone level are more desired and will be used by more ants; therefore, their pheromone level will increase. On the other hand, the pheromone level of routes that are rarely used gradually decreases and eventually vanishes due to evaporation. It has been shown that selecting routes based on the pheromone level results in the selection of shortest routes [39]. The indirect communication mechanism used by ant systems makes them a potential solution for MANET routing problem since they can offer the shortest paths while keeping the routing overhead to a minimum. In the ARA algorithm, the artificial ants are small size packets that contain the source ID, the destination ID, a sequence number, the ID of the transmitting node, and the number of hops traversed so far. When a node requires a route to a particular destination, it creates forward ant (FANT) which is basically a RREQ packet and transmits it to its neighbors. When an intermediate node receives a FANT, it adds a triple of the following format: (destination ID, next hop, pheromone value) to its routing table; here, the destination ID is the ID of the source node contained in the FANT, the next hop is the ID of the neighbor delivered this FANT, and the pheromone value is the number of hops traversed so far. Based on the source ID, destination ID, and the sequence number of the FANT, intermediate nodes can realize the reception of the first copy of a particular FANT. When an intermediate node receives the first copy of a FANT, it increments the FANT's hop count field, sets the FANT's transmitting ID field to its own ID, and retransmits it to all of its neighbors. The effect of sending a FANT is that all nodes will have multiple paths to the source node. When the destination node receives a FANT, it extracts its contents and destroys it; moreover, the destination generates a backward ant (BANT) which has the same structure and propagates by the same mechanism as the FANT; however, the BANT propagates toward the source and results in the creation of multiple routes to the destination. Figures 2.14 and 2.15 explain the propagation of FANT and BANT packets through the network.

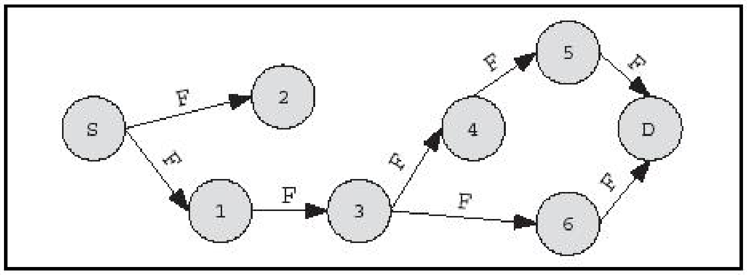

Figure 2.14: FANT propagation toward the destination.

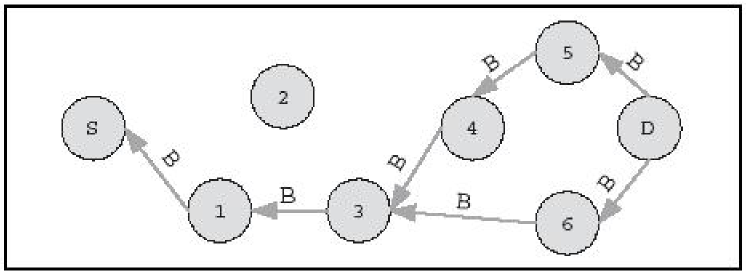

Figure 2.15: BANT propagation toward the source.

Once the route to the destination is established using FANT and BANT packets, data packets are used to maintain this route by incrementing the pheromone values of the links used by data packets during their trip to the destination. When a node discovers a link failure, it rests the pheromone level of that link to zero and looks for alternative route in its routing table; if there is no alternative route, this node sends a *ROUTE ERROR* message along with the data packet to its neighbors expecting that they can relay it to the destination. If all neighbors do not have a valid route; they relay the packet backward toward the source node. If the backtracking process reaches the source node, the source initiates a new route discovery process.

Although the ARA algorithm is able to produce multiple routes to the destination, the route discovery phase uses two flooding processes (FANT and BANT flooding) and this might degrade the scalability potential of ARA in large networks that have heavy traffic.

2.2.11 Flow Oriented Routing Protocol (FORP)

The FORP protocol [86] is an on-demand routing protocol designed to minimize disruptions that result from route failures during communication sessions. The disruption minimization is achieved by employing a mechanism to predict route failures ahead in time so that alternative routes can be found and used before the occurrence of failures. The FORP protocol is well suited for real time traffic; however, it can be used with non real-time traffic. It is assumed that nodes are able to predict the *link expiration time* (LET) of links connecting them to their neighbors by knowing the mobility characteristics of those neighbors; i.e., the speed and the direction of movement. The minimum *link expiration time* of the links of a given route is called the *route expiration time* (RET). When a node needs a route to a given destination node, it initiates a route discovery process using a *FOLW REQ* packet. The route discovery process used by the FORP protocol is very similar to that used by the DSR protocol [61]; however, the FORP protocol introduces the following modifications:

- Before retransmitting the *FLOW REQ* packet, intermediate nodes embed the LET of the link over which the *FLOW REQ* packet was received into this packet.
- Intermediate nodes retransmit the *FLOW REQ* packet if it is the first copy or it includes a route that has a better RET than that of the *FLOW REQ* packet retransmitted most recently.

Routes are selected by destinations based on the RET characteristics: the route with the largest RET is preferred. If there are more than one route with the same RET, the one with the minimum number of hops is selected. After selecting a route, the destination node sends a *FLOW SETUP* packet to the source node using the selected route; all nodes located on the selected route maintain routing information for this route. During data transmission, intermediate nodes embed the LET of the used links into data packets so that the destination is updated about the RET of the selected route. When the selected route is up to expire, the destination creates a *FLOW HANDOFF* packet destined to the source and floods it to the network; the function and the propagation mechanism of the *FLOW HANDOFF* packet are the same as those of the *FLOW REQ* packet. When the source receives the *FLOW HANDOFF* packet, it selects the best route to the destination and sends a *FLOW SETUP* packet to the destination so that nodes along the selected route maintain routing information for it. The main disadvantage of FORP is that it used flooding for route discovery which may result it lack of scalability.

2.2.12 Cluster-Based Routing Protocol (CBRP)

The CBRP protocol [58] is an on-demand routing protocol in which nodes are grouped into clusters to control the flooding process used in route discovery phase. Every node can be in one of three possible states: *cluster head, cluster member, or undecided*, i.e., not affiliated. When a node joins the network it has the undecided state. To affiliate with a cluster, a node in the undecided state, ϑ, broadcasts a *Hello* message and waits for a reply from a cluster head; if a positive reply is received within predefined time period, ϑ changes its state to cluster member. Otherwise, if it has at least one bidirectional link with other cluster members, ϑ changes it state to cluster head. Every node maintains a *neighbor table* in which it stores the ID, state, and link statues (bidirectional or unidirectional) for each neighbor. Neighbor nodes exchange their neighbor tables with each other. The exchange of neighbor tables allows nodes to discover their two-hop neighborhood. Moreover this enables cluster heads to identify member nodes that can be used to reach nodes in other clusters; these nodes are called gateway nodes. The route discovery approach used by the CBRP protocol is the same as that used by the DSR protocol except that the CBRP protocol allows only cluster heads and gateways to participate in the route discovery flood. The CBRP protocol uses the source routing approach in which data packets include a routing header that contains the complete path from source to destination. To avoid using flood process for route repair and to minimize the delay associated with repairs, the CBRP protocol uses local repair mechanism in which the intermediate node whose link fails try to bypass it by using its two-hop topological information. In addition to the cost incurred by cluster formation and maintenance, the inclusion of the whole path in every data packet consumes a large portion of the packet and increases the routing overhead.

2.2.13 Characteristics of the reactive routing protocols

- The main drawback of reactive protocols is the latency associated with route discovery phase.
- Most of the reactive protocols, except CBRP, use pure flooding and hence incur the same cost during the first communication session between any two nodes.

2.3 Hybrid routing protocols

The protocols of this category use a combination of the proactive and reactive routing strategies depending on the destination being sought anticipating that the combination will attain the advantages of both strategies while alleviate their drawbacks. Hybrid routing protocols use proactive routing technique to find routes to nodes located in nearby regions and use reactive routing technique to find routes to nodes in distant regions. Generally speaking, hybrid routing protocols are more scalable than protocols of the proactive and reactive categories.

2.3.1 Zone Routing Protocol (ZRP)

The ZRP protocol [53] introduces the notion of node's routing zone which is the set of nodes within a predefined distance measured in terms of hops; this distance is called the *zone radius* (*rzone*) and it plays an important role in the mechanics of the ZRP protocol. Every node maintains routes to all members of its routing zone using any proactive routing protocol. When a node needs a route to another node that is not a member of its routing zone, it initiates a route discovery process by sending a RREQ packet to the furthest nodes in its routing zone (these nodes are known as the *peripheral nodes*); i.e., routes to remote nodes are created reactively. The RREQ packet contains the source ID, the destination ID, and the maximum number of hops it is allowed to traverse. Since nodes know all members of their routing zones, If the RREQ packet was received by a node that is not a member of the destination's routing zone, that node appends its ID to the RREQ packet, decrements the maximum hop count, and retransmits the packet to its *peripheral nodes*. When the maximum number of hops reaches zero, the RREQ packet is discarded. On the other hand, when the RREQ packet reaches a node that is a member of the destination's routing zone, a RREP packet is generated and sent back to the source using the reverse path of that used by the RREQ packet. Besides the reduction in routing overhead achieved by allowing only *peripheral nodes* to participate in the route discovery process, the route discovery process provides multiple routes which allows for reduction in route reconstructions. The performance of the ZRP protocol is strongly affected by the value of the *rzone* parameter. If *rzone* is set to a small value, the ZRP's performance is similar to that of pure flooding; on the other hand, if *rzone* is set to large value, the ZRP protocol exhibits more proactive behavior.

2.3.2 Zone-based Hierarchical Link State (ZHLS) protocol

The ZHLS protocol [59] is global positioning system-based routing protocol in which nodes are grouped into non-overlapping clusters called zones. Every zone covers a specific geographical region and has a unique ID assigned during design time. Nodes are associated with zones based on their positions. Every node sends periodic *link request* packets to its neighbors who in turn send back *link reply* packets of the form *(Node ID, Zone ID)*. Upon the reception of the link replies from all neighbors, each node recognizes all of its immediate neighbors. Nodes that have neighbors belonging to adjacent zones are called *gateway nodes*. In the ZHLS protocol, Nodes exchange two types of link state packets:

- **Node level link state packets** (node LSPs): Each node creates a node LSP in which it stores the IDs of neighbors belonging to its own zone, and only the zone ID of neighbors belonging to adjacent zones. Node level link state packets are exchanged by nodes of the same zone; nodes discard packets arriving form other zones. Each node creates intra-zone routing table based on the contents of the LSPs received from neighbors.

- **Zone level link state packets** (zone LSPs): These packets are created based on the information contained in the node level LSPs of gateway nodes; namely from the IDs of the adjacent zones that can be reached through gateway nodes. Upon the reception of node LSPs form all members of its own zone, each node of that zone ends up with the same zone LSP packet. These packets propagate through the whole network via the gateway nodes. The exchange of zone level LSPs allows all nodes to create inter-zone routing table.

Data packets forwarding inside the same zone is accomplished through the use of intra-zone routing tables. Forwarding messages between nodes of different zones is achieved by specifying the destination's node ID and zone ID and making use of the inter-zone and intra-zone routing tables. To determine the zone ID of the destination node, the source node, ϑ, sends a *location-request* packet that has the following structure: (Source-Node-ID, Source-Zone-ID, Destination-Node-ID, X), where X implies that this packet has to be sent to all zones using the shortest paths according to the inter-zone routing table; upon the reception of this message, each gateway looks for the destination into its intra-zone routing table; if the destination is not a member of the zone, i.e., it is not included in the intra-zone routing table, the request is forwarded to other zones; otherwise, a location reply of the following format: (Destination-Node-ID, Destination-Zone-ID, Source-Node-ID, Source-Zone-ID) is sent back to the source node. The location request process incurs much less overhead than that results form flooding process used by most reactive protocols. The main drawback of the ZHLR protocol is that it requires a static map to assign nodes to zones based on their positions; this map is not always available especially when the network boundaries change dynamically.

2.3.3 Scalable Location Update Routing Protocol (SLURP)

The SLURP protocol [90] is another routing protocol that uses location information to accomplish the routing task; however, the SLURP protocol needs only approximate location information to deliver messages. The SLURP protocol is based on a combination of two procedures: the geographic location management procedure and the Most Forward with fixed Radius without backward progression (MFR) geographic routing procedure [54]. In the MFR procedure, based on the location information of the destination, a neighbor is chosen as the next hop to the destination if and only if it is the farthest neighbor in the direction of the destination. The originality of the SLURP protocol resides in its location discovery mechanism. In this mechanism, the network geographical span is divided into non-overlapping rectangular sub-regions. Each sub-region is completely identified by the coordinates of its bottom left and upper right corners. A static mapping function is used to assign each node, ϑ, to a sub-region based on $\vartheta's$ ID, this region is known as $\vartheta's$

home-region. The mapping function is designed such that nodes are assigned evenly to home-regions. Any node present at a given sub-region, which is not necessarily its home-region, is responsible for storing the location information of all nodes assigned to that home-region. Every node updates nodes present at its home-region about the current sub-region it is located in. A location update is triggered every time a node moves form one sub-region into another one. Movements inside the same sub-region trigger no location update. Nodes maintain location caches in which they store the node ID, location coordinates, current sub-region ID, and the neighbor to be used as a next hop to that node. When a node needs to send a message to any destination, it first needs to find the current sub-region of that destination; If this information is not available in the source's location cache, the source initiates a *location discovery* process by sending a location-discovery packet composed of the source ID, the source's current sub-region, the destination ID, a sequence number, and a search level field (the default search level is 1). The sequence number is used to realize queries that were seen before, and the search level is used to specify whether only nodes present in the destination's home-region are allowed to reply (search level=1) or even nodes in the sub-regions that are adjacent to the destination's home-region are allowed to reply when the later is empty (search level=2). The source node sends the location-discovery packet to all its neighbors. The source's neighbors use the location-discovery packet to update the source's location information in their caches; only one neighbor (according to the MFR procedure) is allowed to retransmit the location discovery packet. The same process of receive-update-retransmit is repeated until the destination's home-region, or an adjacent sub-region if the later is empty, is reached. In the destination's home-region, the first node to receive the location-discovery packet generates a route reply packet and sends it back to the source node. Once the destination's current sub-region becomes available to the source node, data packets are forwarded in the direction of the destination's current sub-region using the MFR procedure. Upon the arrival of data packets at the destination's current sub-region, any source routing protocol can be used to deliver the packets to the destination. The designers of the SLURP protocol used the DSR protocol for intra sub-region routing purposes. The main disadvantage of SLURP is its use of static sub-region location information which might not be available at design time.

2.3.4 Distributed Spanning Trees based routing protocol (DST)

The DST protocol [81] exploits the fact that even highly dynamic ad hoc networks exhibit some topological stability at different times or at different regions at the same time; this fact is exploited by using spanning trees in regions that exhibit some topological stability and employing controlled flooding in regions that show rapid topological changes. Moreover, DST introduces the concept of *connectivity through time* to deliver messages between end points that might not be connected at any time; in such cases, message delivery is achieved by having the intermediate nodes to store data packets sent by the source node expecting that these nodes might be connected to the destination node soon; if no route is offered during specified period of time, intermediate nodes drop the stored data packets. DST organizes the network nodes into a set of dynamically created trees. Every tree is identified and controlled by its *root node*. Every node in the tree keeps track of its parent's ID, its children IDs, and the ID of the root node (to prevent the creation of cycles). Neighboring nodes that belong to different trees are called *contact nodes*. When two trees come in contact with each other, they try to merge together into one tree and one of the roots of the original trees gives up its role and becomes an ordinary node. In cases where the neighboring tree is likely to move out of the communication range shortly, no attempt is made to merge with this tree. A bridge is established between the contact nodes of the neighboring trees. At any time, the DST protocol considers the whole network as a forest of spanning trees. Nodes of the same tree know the IDs of each other and maintain routes to each other in proactive manner. Routing between nodes of different trees is achieved reactively using one of two mechanisms:

- **Hybrid Tree-Flooding (HTF)**: packets are sent to all neighbors in the tree and to all bridges leading to neighboring trees. In addition to retransmitting the received packet, every node stores a copy of the packet for a *holding time* period. If a bridge to a new neighboring tree is established during the holding time, the buffered packet will be forwarded to the new neighboring tree. Otherwise, the buffered packet will be dropped.
- **Distributed Spanning Tree Shuttling (DST)**: packets are sent by the source node to all nodes and bridges down the tree. When a packet reaches a leaf node, i.e., a node that has no children, it is sent up the tree to a certain height called the *shuttling height*. After reaching the shuttling height, the

packet is retransmitted to all nodes and bridges down the tree.

The reliance of the DST protocol on root nodes to control the trees makes it less robust since root nodes become potential points of failure.

2.3.5 Distributed Dynamic Routing (DDR) protocol

In the DDR protocol [74], nodes are grouped into dynamically created trees, i.e., zones. Theses trees differ from those used in the DST protocol in that they do not have root nodes; therefore, the DDR protoco is more robust than the DST protocol. Each node is a member of only one tree; hence, the ad hoc network can be thought of as a set of non-overlapping trees, i.e., zones. The number of neighbors of a given node is known as the node's degree. Every node declares one of its neighbors as a *preferred neighbor*. The neighbor with the highest degree is selected as preferred neighbor; if there is more than one neighbor with the same degree, the one with the larger ID is selected as the preferred neighbor. Every node has only one preferred neighbor; however, the same node can be selected as a preferred neighbor by many nodes. Nodes of the same neighborhood know the IDs and the degrees of each other through the exchange of periodic beacon messages that contain five fields: the node's ID, the ID of the tree to which the node is belonging, the node's degree, a flag field, and the ID of the preferred neighbor. The frequency of beacon messages depends on the rate of node mobility. The flag field is used to distinguish between two types of beacon messages:

- **Preferred neighbor declaration message**. In this message, the preferred neighbor field contains the ID of the neighbor selected as the preferred one. The flag field is set to 1 in this type of beacon messages.
- **Neighborhood change notification message**. In this message, the preferred neighbor field contains the ID(s) of new or missed neighborhood members. The flag field is set to 0 in this type of beacon messages.

Once the preferred neighbor is selected, it is declared to all neighbors by sending a beacon message of the first type. Only the preferred neighbor has the privilege to retransmit this message. Trees are created by connecting each node to its preferred neighbor. Neighboring nodes that belong to different trees are called gateway nodes. Figure 2.16 illustrates the selection of preferred neighbors and the creation of trees.

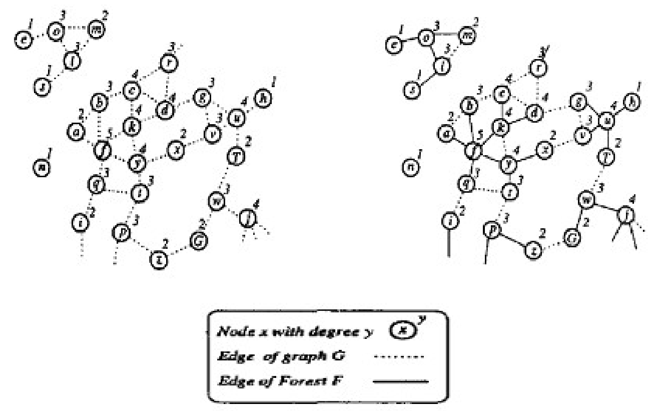

Figure 2.16: Network before tree creation (left), after tree creation (right).

Application
HARP
DDR
Network

Table 2.1: Layered view of HARP and DDR

Using the information contained in the beacon messages, every node of the network creates two tables:

- **Intra-zone table:** this table contains an entry for every neighbor wherein the neighbor's ID along with the IDs of all nodes that are reachable trough that neighbor are stored.
- **Inter-zone table:** this table contains an entry for each neighboring zone; this entry stores the zone ID, the ID of the gateway node to this zone, and the stability of the connection with this zone.

Routing within the same zone is accomplished by consulting the intra-zone routing table to find the next hope toward the destination. Inter-zone routing was not discussed in detail in [74]. The designers of the DDR protocol introduced the HARP protocol [73] as an extension that uses the inter-zone tables produced by the DDR protocol to perform the inter-zone routing task. Table 2.1 illustrates the proposed integration of the HARP and the DDR protocols.

The HARP protocol performs the inter-zone routing reactively by initiating route discovery process through the generation and transmission of route request packet (RREQ) every time a route to another zone is needed. The source node uses its intra-zone routing table to forward the RREQ packet to the gateway nodes of its tree, i.e., zone. The RREQ packet contains a field that shows the stability of the path traversed so far. This field is initiated to infinity by the source node. Upon the reception of the RREQ, each gateway compares the value of the RREQ's stability field with that of its connection with the neighboring zone; the smaller of these two values is assigned to the stability field of the RREQ. After updating the stability field, gateway nodes forward the RREQ to their neighboring zones. Any other copy of the same RREQ is ignored unless it contains a higher stability. This process is repeated by all intermediate zones until the destination zone is reached. Within the destination zone, the RREQ packet is forwarded using the intra-zone routing table. After receiving the RREQ packet, the destination node creates a route reply packet (RREP) and sends it back to the source node using the same route traversed by the RREQ packet. If multiple RREQ packets were received by the destination, i.e., multiple routes are available, the most stable route is selected. Once the RREP packet reaches the source node, data transmission can be started immediately using the discovered path. If a route failure occurs during an active session, an error message is sent to the source urging it to start a new route discovery process. Route discovery is performed periodically during the life time of an active session to find a more stable route, if any, even if the used route is still valid. This proactive route maintenance process aims at avoiding route recovery operations and at reducing the associated delay. The DDR and the HARP protocols reduce the routing overhead by minimizing the number of nodes involved in the route discovery process thanks to the intra-zone routing tables. However, the preferred nodes used in the intra-zone routing mechanism are potential points of failure especially if a node is selected by many others as their preferred neighbor.

2.3.6 Characteristics of the hybrid routing protocols

- Hybrid routing protocols have higher scalability potentials than pure reactive and proactive protocols due to the fact that they use some sort of control structures (zones, clusters, or trees) that can accommodate large number of nodes.
- Hybrid routing protocols find routes faster than reactive protocols using less overhead compared to that of proactive protocols, i.e., they strike a balance between delay and routing overhead.

- Hybrid routing protocols are more robust in the sense that topology changes are handled transparently within the affected zone in a localized manner without affecting routes in other zones.

2.4 Summary

This chapter surveyed the most widely renowned routing protocols and highlighted their advantages and disadvantages. It is apparent that routing protocols in which all nodes take part in routing at all times, a.k.a. ***flat routing protocols***, have very poor scalability potential. On the other hand, routing protocols in which nodes are grouped into clusters and a small set of nodes from each cluster are elected to perform and supervise routing, a.k.a. ***hierarchal routing protocols***, have more scalability potential that results from their use of topology abstraction and from the localization of the effects of route failures.

The literature is rich of protocols proposed for MANETs clustering. Most of these protocols are based on some kind of graph domination. The next chapter provides a review of the most cited MANETs' clustering protocols.

Chapter 3

Clustering

Clustering in MANETs' is the process of partitioning mobile nodes are partitioned into groups in which only few elected nodes known as *cluster-heads* or *cluster-leaders* are allowed to participate in routing. However, not all MANETs' clustering algorithms use cluster-heads. Some clustering algorithms, the ones proposed in [88] and [68] for example, partition mobile nodes into clusters in which members of the same cluster maintain single hop or multi hop routes to each other and use *gateway* nodes, i.e., nodes that have links to members of neighboring clusters, to forward messages to those clusters, Figure 3.1 shows an example of clusters with cluster-heads and Figure 3.2 shows clusters with no cluster-heads.

In addition to increasing MANETs' scalability by reducing the number of nodes generating and storing routing information, clustering allows for spatial reuse of resources among different clusters[65], makes a large MANET seem smaller by representing it as a set of interconnected clusters, and makes a highly dynamic MANET seem less dynamic by increasing cluster stability[42]. On the other hand, clustering has its own overhead associated with cluster formation and maintenance. In order to get the most out of clustering, the last two decades have seen massive amount of research work aiming at devising fast, light (in terms of signaling overhead), and stable clustering algorithms [15]. The following section gives a review of the previous research work in this field.

3.1 previous work

The main task in clustering is to affiliate every node ϑ with at least one cluster. This affiliation guarantees coverage for all nodes. In cluster-head based clustering,

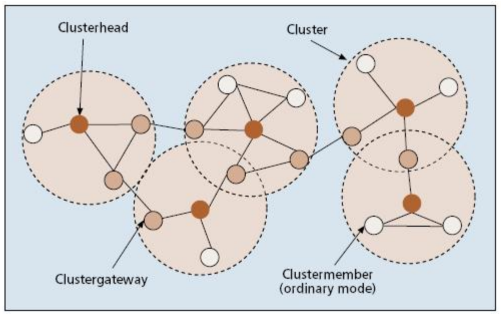

Figure 3.1: clustering using cluster heads.

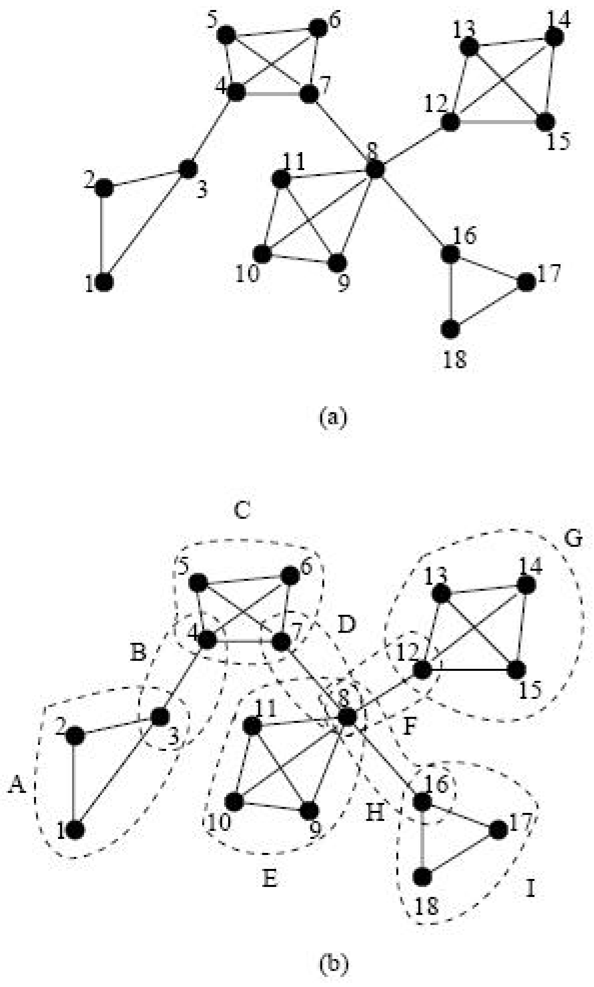

Figure 3.2: clustering using no cluster heads.

node ϑ belonging to cluster C is said to be dominated by the head of cluster C; hence, the notion of graph domination or one of its variants is the basis of the majority of MANETs clustering algorithms [27]. MANETs' topology is treated as undirected graph, $G = (V, E)$, where the set of vertices V represents the set of mobile nodes, and the set of edges E represents the wireless links.

A set $S \subset V$ is called a *distance-k* **dominating set** of a graph $G = (V, E)$ **iff** $\forall\ \upsilon \in V \Rightarrow \exists\ \gamma \in S$ such that $\upsilon \in [\gamma]_k$; where $[\gamma]_k$ is the *k-hop* neighborhood of γ. A vertex $\vartheta \in$ S is said to dominate all its *k-hop* neighbors in $V - S$. An edge is *dominated* if at least one of its end points is in S; otherwise; it is *free*.

A dominating set S is an **independent dominating set iff** $\forall\ \upsilon,\ \gamma\ \in S \Rightarrow \upsilon\ and\ \gamma$ are not adjacent to each other. If the induced subgraph $\langle S \rangle$ of a dominating set S is connected, S is called a connected dominating set, CDS. On the other hand, if the subgraph weakly induced [1] by S is connected, S is called a weakly-connected dominating set, WCDS. Figures 3.3, 3.4, 3.5, and 3.6 show examples of the DS, CDS, WCDS, and the weakly induced graph, respectively.

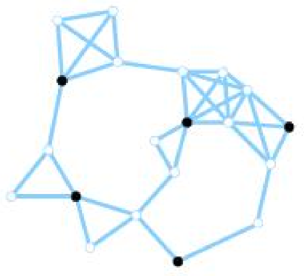
Figure 3.3: DS.

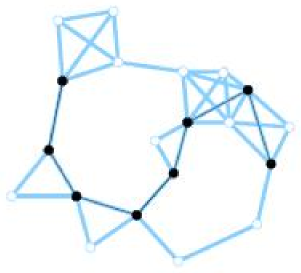
Figure 3.4: CDS.

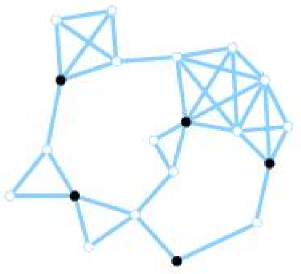
Figure 3.5: WCDS.

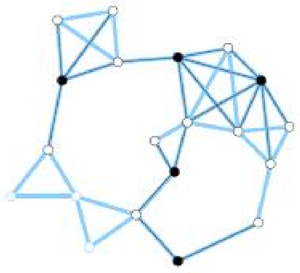
Figure 3.6: Weakly Induced Graph.

[1]A subgraph induced by a set of vertices S is weakly induced subgraph , $\langle S \rangle_w$, if it contains all vertices in S, their neighbors, and all the dominated edges.

Minimizing the cardinality of the dominating set produces a simpler abstracted topology of the MANET and allows for using shorter routes in the clustered topology. Since its NP-complete to find the minimum connected dominating set (MCDS), researchers resorted to approximation algorithms and heuristics to tackle this problem [89]. The following subsections give a description of graph domination-based MANET clustering algorithms.

3.1.1 Clustering using independent dominating sets

Algorithms based on independent dominating sets prohibit adjacency between cluster-heads. The linked cluster algorithm (LCA), the earliest clustering algorithm proposed in the literature, falls in this category [12]. In the LCA algorithm, the selection of cluster-heads is based on node IDs, a node ϑ is elected as a cluster-head if it has the highest ID in its neighborhood. Nodes that are adjacent to more than one cluster-head are called *gateways*, the rest are ordinary nodes. In [49], Gerla and Tsai presented two clustering algorithms that produce independent dominating sets. One algorithm elects the node with the smallest ID in its neighborhood for the role of cluster-head. In the other algorithm, the notion of *node degree* which refers to the number of node's neighbors is used to elect cluster-heads. The node that has the highest degree, i.e., the largest number of neighbors, is elected as the cluster-head; in a case of tie, the one with the smaller ID is elected. Based on the performance evaluation presented in [31], the *highest degree* algorithm has a lower stability compared to the *lowest ID* algorithm; the reason for the low stability of the *highest degree* algorithm is the sensitivity of node degree to topology changes. Lian and Hass used the *k-hop degree* to elect cluster-heads [64]. In [23], The *lowestID* and *highest degree* of Gerla and Tsai are generalized by Chen et al. to the *K-lowestID* and *k-CONID* in which cluster-heads are elected among nodes of *k-hop* neighborhoods based on the ID in the first and on the connectivity (with resorting to the ID for tie breaking) in the second. In [15], Basagni proposed that cluster-heads can be elected based on weights that are assigned to nodes according to any meaningful measure. For example, weights can be assigned according to node ID, node degree, or a mobility related parameters. A node is elected as a cluster-head if it has the largest (or smallest) weight in its neighborhood. In [16], the signal strength of any two consecutive *hello* messages received from a given neighbor node is used to evaluate the relative mobility with respect to this neighbor. Every node computes the mean of the squares, i.e., the variance with respect to zero, of its relative mobilities with respect to its neighbors and advertise this value as its weight. The node that has the smallest weight among its neighbors is elected as a *cluster-head*. Er and Seah [42], used a similar concept to build 2-hop clusters. A group of 2-hop clusters can merge to form D-hop clusters if they exhibit similar mobility behavior. In [10], Beongku and Papavassiliou proposed a clustering algorithm that assigns weights based on node mobility and node ID. Mobility information is assumed to be available through the location positioning system GPS. Every node exchanges its velocity information with its neighbors. Nodes whose aggregate mobility with respect to their neighbors are less than a certain limit are considered a cluster-head candidates. The candidate that has the smallest ID becomes the *cluster-head*. If two cluster-heads with a comparable mobility come into communication range of each other, their clusters are merged in one. The Clusters produced by this algorithm are not necessarily of the same size. In [32], Based on the knowledge of neighbors' velocities, every node uses fuzzy inference to predict the number of neighbors that stay within its transmission range for a certain period of time δ and advertises this number to its neighbors as its $\delta-degree$. The neighbor with the largest $\delta-degree$ is elected as a cluster-head. In [21], Chatterjee et al. used the weighted sum of four parameters to assign weights to nodes. These parameters are: the absolute difference between the node's degree and an optimal degree (Δ_v), the sum of distances between the node and its neighbors (D_v), node's speed (M_v), and the length of the time period during which the node served as a cluster-head (P_v). The node that has the smallest weight among its neighbors is elected for the cluster-head role. The *Passive clustering* algorithm [95] attempts to eliminate clustering overhead by using the on going data traffic to establish and maintain the clustered architecture in an on-demand basis. The First-Declaring-Wins rule is used to elect cluster heads. A node that is not affiliated with a cluster is said to be in the *undecided state*. Initially, all nodes are undecided. A node that has data to send and has no cluster-head neighbor, declares itself a cluster-head. Once a cluster-head is declared, no neighbor is eligible to claim the cluster-head role until the current cluster-head resigns or moves out of sight. All neighbors of the declared cluster-head become cluster members. Initially, all cluster members are allowed to rebroadcast messages received from the cluster-head, and eventually only gateways and cluster-heads are allowed to retransmit. A node that receives data messages only from cluster member neighbors and has no cluster-head neighbors claims the cluster-head role and the cluster member neighbor that delivered the first data message from his cluster-head to the recently declared cluster-head becomes a gateway node between the two clusters. This process continues until the message reaches its destination.

The main disadvantage of clustering algorithms based on independent dominating sets is that when two cluster-heads move into the transmission range of each other, one of them relinquishes its role and reclustering is performed to maintain the clustered architecture; this reclustering effect may spread throughout the whole network in a *chain reaction-like* process [48].

3.1.2 Clustering using dominating sets

As a part of the *CEDAR* routing protocol [84], Sivakumar et al. proposed a simple heuristic to approximate the MDS of a MANET. The nodes in the MDS constitute the cluster-heads of the clustered architecture. Every node u computes its degree ($d(u)$) and its effective degree ($d^*(u)$) which is equal to the number of nodes dominated by u, and advertises these values to its neighbors in the form of the pair $< d^*(u), d(u) >$. The pairs received from neighbors are ranked in lexicographical order. The node with the largest pair in u's neighborhood is elected as u's *Dominator*, ties are broken using node ID. Each dominator sets its effective degree to the number of its *Dominatees* and declares itself a cluster-head. Initially, all nodes have *zero* effective degree; hence, at the beginning, the node with the highest degree is elected as cluster-head; after that, the clusters stability is improved by giving cluster-heads more chance to be re-elected since they have higher effective degrees than ordinary nodes. Increasing the cluster span, i.e., the maximum number of hops separating a cluster-head and a cluster member, reduces the number of clusters and improves the network scalability; however, it increases the cluster-heads loading. The Max-Min clustering algorithm [9] is designed to strike a balance between cluster span and cluster-head loading. In this algorithm, every node elects the node with the largest ID in its *d-hop* neighborhood for the cluster-head role. Once informed about its election, the elected node assumes its cluster-head role even if its ID is not the largest in its own *d-hop* neighborhood. By permitting more nodes within the same neighborhood to be cluster-heads, Max-Min produces stable *d-hop* clusters that have almost the same number of nodes. As a result, The Max-Min clustering algorithm allows for load balancing; however, it incurs high communication overhead [60]. Belding-Royer proposed two clustering algorithms in which the principle of First-Declaring-Wins is used to elect cluster-heads. The first algorithm (the ARC algorithm) builds a one-level clustering hierarchy while the second algorithm (The ARCH algorithm) generalizes the method of the ARC algorithm to produce a multi-level hierarchy in which the number of levels adapts to network density and nodes' mobility behavior [17]. To reduce the number of role switching and hence reduce the cluster maintenance overhead, the ARC and ARCH algorithms force any cluster-head to relinquish its role only if its entire cluster becomes a subset of another cluster. Jia et al. [57] used randomization to speed up the running time of the clustering algorithm presented in [64]. They called the new algorithm the Local Randomized Greedy algorithm *LRG*. While the deterministic version of this algorithm (presented in [64]) runs in linear time that is a function of the number of nodes, the LRG algorithm runs in a polylogarithmic time with high probability. However, the LRG algorithm exhibits higher message complexity and in some cases produces dominating sets of larger sizes compared to those produced by its deterministic version. The LRG algorithm proceeds in rounds each of which contains the following steps:

- Every node v calculates its span ($d(v)$), a.k.a. node coverage ($c(v)$), which is equal to the number of its uncovered neighbors including itself if it is not covered; then, every node declares its rounded span which is equal to $\lfloor \log {}_{b}^{d(v)} \rfloor$, where b is a constant.
- A node, v, considers itself as a candidate if its rounded span is equal to largest rounded span in its *2-hop* neighborhood.
- Every uncovered node u reports to its neighbors the number of candidates adjacent to it, this number is called the node support $s(u)$.
- A candidate, v, is added to the dominating set with a probability

$$P(v) = \frac{1}{med(v)}$$

where $med(v)$ is the median support of the nodes in $c(v)$.

Clustering algorithms based on dominating sets produce more stable clusters, generate less cluster maintenance overhead, and eliminate the chain reaction effect. However, because they allow cluster-heads to be adjacent to each other, they are not suitable for spatial resource reuse.

3.1.3 Clustering using connected dominating sets

In ad hoc networks, a connected dominating set provides for the creation of a *virtual backbone*, a.k.a. spine, that can be used for routing and control purposes [43].

Constructing the minimum connected dominating set (*MCDS*) in general graphs is known to be NP-complete problem. Khular and Guha [50] proposed two centralized approximation algorithms for the MCDS problem. The first algorithm starts by adding the node with he highest degree to the CDS and proceeds iteratively by adding a node or two nodes (whichever leads to the maximum increase in the number of dominated nodes) in each iteration. The second algorithm starts by creating a dominating set and then proceeds by adding nodes to connect the dominating set. An other centralized algorithm with a constant approximation factor for approximating the connected dominating set in a *unit disk graph* (**UDG**) is presented by Marathe et al. [66]. This algorithm starts by creating a breadth-first spanning tree, T, rooted at an arbitrarily selected node v. Nodes are assigned to groups according to their hop distance from the root v; every group has a level value that is equal to its distance from the root. Initially, the root of T is the only member of the CDS. The other members of the CDS are selected as follows:

```
- Set $L$ to the depth of $T$, Let $S_k$ represent the group of nodes at level $K$
- For $K = 1$ to $L$ Do
    - Remove all nodes $u \in S_k$ that are dominated by nodes in $S_{k-1}$.
    - Construct a maximal independent set $MIS_k$ from the remaining nodes
      in $S_k$ and add all members of $MIS_k$ to the CDS.
    - Add to the CDS all nodes in $S_{k-1}$ that have children in $MIS_k$.
```

In [37], Das et al. proposed a distributed version of Khular and Guha algorithms. Being of a distributed nature, Das's implementation of Khular and Guha algorithms is suitable for ad hoc networks paradigm; however, it is characterized by high message and time complexities due to its use of global knowledge about node degrees and also because of its sequential nature. Moreover, the algorithms in [37] do not provide any mechanism by which nodes recognize the completion of one phase of the algorithm and the beginning of the next one. Wu and Li [92] tried to reduce message and time complexities by relying only on knowledge of *2-hop* neighborhood. Starting with an empty CDS, a node ϑ is added to the CDS if it has at least two non-adjacent neighbors. Wu and Li call CDS' members *Gateway Hosts.* In [92] Wu and Li prove the correctness of this simple algorithm; However, they admit that in some cases their algorithm produces trivial CDS that includes all the vertices of the network. in [15], Basagni et. al reported that the CDS created by this algorithm has a large size compared to those produced by other approximation algorithms [15]. In [85], Stojmenovic et al. presented a modified version of Wu and Li algorithm. According to [15], Stojmenovic variant is faster and generates less overhead than the original Wu and Li algorithm; moreover, the CDS produced by Stojmonovic version is smaller in size, and denser; however, it is less robust. Wu and Li algorithm and its Stojmonevic variant are studied in more detail in the next chapter. In the *Span* algorithm [22], CDS members, called *coordinators* in this algorithm, are elected using a similar principle to that used by Wu and Li algorithm [92], i.e., node v is *eligible* for declaring itself a *coordinator* **iff** it has at least two neighbors u and w that are not connected through one or two *coordinators*. To avoid contentions that arise when more than one *eligible* node can connect the same pair(s) of unconnected neighbors (u,w), eligible node v delays its declaration for a back-off time that is directly proportional to v's remaining energy (expressed as a fraction of its maximum value) and the number of pairs that are connected through v. During the back-off time, if no node declares itself as a *coordinator*, v sends its declaration; otherwise, v has to reevaluate its eligibility based on the new conditions. In [80], Qayyum et al. used the concept of *Multipoint Relays* to design an efficient broadcasting algorithm that reduces the number of retransmitting nodes while guaranteeing the delivery of broadcast messages to all network nodes. In [2], Adjih et al. modified the work presented in [80] in such a way that the new algorithm produces a *source-independent MPR* called the *MPR-CDS*. The authors of [2] the correctness of this algorithm. However, the approximation factor of this algorithm is inferior to that of Wu and Li algorithm. In [91], Wu modified the work presented in [2] and in [80] in such a way that reduces the size of the produced *MPR-CDS* and hence improves its approximation factor. According to [15] the modified *MPR-CDS* is faster than Wu and Li algorithm. In [93], Wu et al. used the complete *2-hop* neighborhood information to reduce the size of the CDS produced by the algorithm presented in [91]. The authors of [93] call the new algorithm *EEMPR* to emphasize the fact that it is an *extension* to the *enhanced MPR* presented in [91]. The *MPR-CDS* and the its variant presented in [91] are studied in more detail in the next chapter. In [41], Dubashi et al. proposed two distributed algorithms that produce a connected dominating set with a running time that is polylogarithmic in the number of nodes. Both algorithms start by creating a dominating set S; the *LRG* algorithm [57] is used for this task. Once the dominating set S is created, a connected subgraph (H) that spans $G^3[S]$ is used to connect S, where $G^3[S]$ is the third power of G [2]. Since the length of a path connecting

[2]The power of a graph: Given a graph $G = (V, E)$, the i^{th} power of G, G^i, is a graph in which there is an edge $e = (u, v)$ between any pair of nodes (u, v) **iff** $d_G(u, v) \leq i$

any two members $u, v \in S$ is at most three hops, every edge $e = (u, v) \in H$ adds at most two new nodes w, z to S; therefore, minimizing the number of edges in H minimizes the size of the created CDS. Inspired by the following graph theoretic lemma [3]:

A simple undirected graph on n vertices has at most $n^{1+\frac{2}{g-1}} + n$ edges, where g is the length of the smallest circle in the graph.

in the two algorithms presented in [41], all cycles of length up to $\lfloor 1 + 2\log|S| \rfloor$ are destroyed by deleting an edge of every such circle. The edges to be deleted are selected randomly in one algorithm and deterministically in the other one. The authors of [41] proved that both algorithms produce a CDS in polylogarithmic time. Alzoubi et al. proposed many distributed approximation algorithms to produce a connected dominating set for a unit disk graph [5, 7]. All these algorithms start by creating a maximal independent set (MIS) and then nodes are added to connect the MIS. In [7], The MIS creation phase starts by a ranking stage in which every node v is assigned a unique rank. Ranks are based on node level and node ID. The rank of node v is given by $(Level(v), ID(v))$. To determine the node level, an arbitrary spanning tree (T) is constructed. The root of T is elected using the *leader election* algorithm presented in [33]. The level of node v is set to the number of hops in T form the root to v. After completing the ranking stage, the MIS creation proceeds as follows:

- Initially all nodes are colored *White*.
- A node v declares itself a *Dominator* and changes its color to *Black* **iff** it has the lowest rank among all its neighbors. All neighbors of v change their colors to *Gray* and declare themselves as *Dominitee* nodes.
- If all lower-rank neighbors of a white node u have declared themselves dominatees, node u declares itself a *Dominator* and changes its color to *Black*.

The MIS creation phase terminates when all nodes are colored either *Gray* or *Black*. The MIS is composed of all the *black nodes*. It is proven in [7] that the MIS created by this algorithm satisfies the following property:

Any set $U \subset MIS$ is 2-hops away from its complement set $V = MIS - U$

[3]Lemma 15.3.2 [67]

In the final phase of the algorithm a tree that spans all the *black nodes* is constructed, the authors of [7] call it the *Dominating Tree.* Initially, the *Dominating Tree* is empty. The root of T is the first *black node* to be added to the *Dominating Tree.* Whenever a *black node* v is added to the *Dominating Tree*, v sends an *invitation message* to all its neighboring *black nodes* that are at most two hops away. Whenever a *black node* that is not a member of the *Dominating Tree* receives an *invitation message* for the first time, it sends a *join* message to the *gray node* from which the first *invitation message* was received. Whenever a *gray node* receives a *join* message addressed to it from one of its children in T, the *gray node* changes its color to *black* and it is added to the CDS. Eventually the *black nodes* of the *Dominating Tree* will form a CDS. The authors of [7] proved that the approximation factor of this algorithm is constant and it is equal to 8. Four variants of this algorithm are proposed in [5, 6, 89]. The first variant is composed of three phases: *leader election phase, MIS construction phase, and Dominating Tree creation phase.* In the *leader election phase* a leader is elected using the algorithm in [33]. Once the leader is elected, the *ranking stage* of the *MIS construction phase* starts. In this variant, ranks are based on node ID; a node's rank is equal to its ID. The rest of this phase is exactly the same as in the original algorithm. However, as a result of the ranking scheme adopted by this variant, the resulting MIS differs from that produced by the original algorithm is that:

Any set $U \subset MIS$ *is at most 3-hops away from set* $V = MIS - U$.

This means that the MIS produced in this variant is sparser than that of the original algorithm; hence, the *gray nodes* that are needed to connect the MIS are more than those added in the original algorithm. After completing the *MIS construction phase*, the leader elected in the first phase starts the final phase by electing a root for the *Dominating Tree.* If the leader is a *black node*, the leader will declare itself as the root of the *Dominating Tree*; Otherwise, the leader selects one of its *black* neighbors to be the root of the *Dominating Tree.* Once the root is declared, it starts the *Dominating Tree* creation procedure that is very similar to the one used in the original algorithm; the only difference between the two procedures is that the *invitation messages* used in this variant can reach all nodes in the *3-hop* neighborhood, while in the old variant *invitation messages* can only reach nodes in the *2-hop* neighborhood. The authors of [5] proved the correctness of this variant in showed that it has a constant approximation factor that is equal to 12; therefore, in terms of performance, the original algorithm outperforms this new variant since it has 8-approximation factor with the same time and message complexities. The second variant is presented in [5] and it is composed of three phases: *leader election phase, ranking phase, and color markup phase.* The leader election and the ranking phases are exactly the same as those used in the original algorithm presented in [7]. The color markup phase is as follows:

- Initially all nodes are colored *White*.
- A node, v, declares itself a *Dominator* and changes its color to *Black* **iff** it has the lowest rank among all its neighbors. All neighbors of v change their colors to *Gray* and declare themselves *Dominitee* nodes.
- If all lower-rank neighbors of a *white* node, u, have declared themselves dominatees, u declares itself a *Dominator* and changes its color to *Black*.
- A *Dominitee* node, w, changes its color from *Gray* to *Black* **iff** one of its children in (T) declared itself as a *Dominator* provided that this child has never declared itself as a *Dominitee* in the past.

Figure 3.7 shows the details of the marking process of this variant. The approximation factor and the time and message complexities of this variant are exactly the same as those of the variant presented in [7]. The third variant of this algorithm is presented in [89] and it is composed of three phases: *leader election phase, MIS construction phase, and Dominating Tree creation phase.* The first two phases are exactly the same as in the original algorithm; however, the third phase is slightly different. The leader elected in the first phase starts the creation of the *Dominating Tree* (T^*) by electing its *gray neighbor* who has the largest number of *black neighbors* to become the root of T^*. Initially T^* is empty and eventually all *black nodes* and all *gray nodes* will join T^*. However, only few members of T^* known as the *internal nodes* are members of the CDS. The *Dominating Tree creation* proceeds as follows:

- The root elected by the leader broadcasts *INVITE2* message to all of its neighbors; then, the root joins T^* .
- Whenever a *black node* receives *INVITE2* message for the first time, it sends *JOIN* message to the sender of the *INVITE2* message, broadcasts *INVITE1* message to all of its neighbors, and joins T^*.
- Whenever a *gray node* receives *INVITE1* message for the first time, it sends *JOIN* message to the sender of the *INVITE1* message, broadcasts *INVITE2* message to all of its neighbors, and joins T^*.

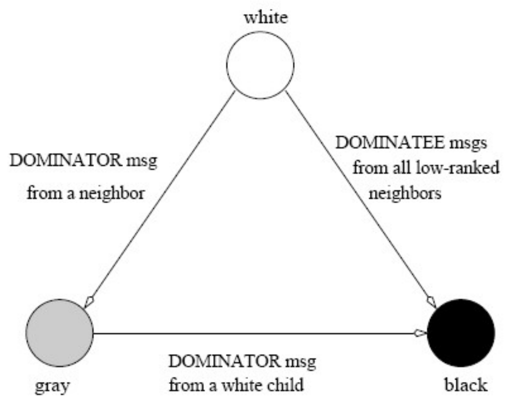
white
DOMINATOR msg from a neighbor
DOMINATEE msgs from all low-ranked neighbors
gray
DOMINATOR msg from a white child
black

Figure 3.7: The markup process of the second variant of Alzoubi algorithm.

- Whenever a node, v, receives *JOIN* message addressed to itself, it becomes an *internal node* of T^*, i.e., a member of the CDS.

As shown in [89] this variant has the same time and message complexities as the other variants; however, it outperforms them in terms of the CDS size. The variant of Alzoubi et al. algorithm that is presented in [6] differs form all other variants in that it does not use any spanning tree. As a result, this variant has lower communication overhead than all other variants; however, its approximation factor is inferior to other variants and its equal to 192. In this variant, the CDS is created in two phases: In the first phase, a *MIS* set is created using the ID-based ranking scheme. Once the *MIS* is created any node, v, is either a *Dominator* if it is a member of the *MIS* or a *Dominatee* if it has at least one neighbor the is a member of the *MIS*. In the second phase of this variant, every Dominator, v, selects a path, p, to all its *2-hop* and *3-hop* Dominator neighbors that has smaller IDs than its own ID. The intermediate nodes of each path p, at most 3 Dominatees, are added to the set of *Connectors* C. The CDS is composed of all nodes $u \in MIS \cup C$. Two randomized distributed algorithms for constructing a CDS in a unit disk graph are presented by Parthasarathy and Gandhi in [75]. Both of these algorithms use a maximal independent set MIS to construct the CDS; hence, they have a constant approximation ratio. These algorithms are faster than Alzoubi algorithm and its variants; however, they are inferior in terms of message complexity. An advantage of these algorithms is that they are designed in such a way that prevents collisions [4] during the MIS and CDS construction phases; hence, their theoretical performance measures reflect the practical ones much more accuratly than other

[4]a collision at node v occurs when two of its neighbors u,w try to transmit messages to v at the same time.

algorithms. On the other hand, these algorithms assume that good estimates of the maximum node degree (Δ) and the number of nodes (n) are available to all nodes of the network, which is an unrealistic assumption. The first algorithm presented in [75] is composed of three phases. In the first phase a D2-coloring [5]of the nodes is performed. A maximal independent set MIS is constructed in the second phase and it is connected in the third phase. The second phase is guaranteed to be completed within C time steps, where C is the number of colors used in the first phase. At every time step, only nodes of a given *Color Class c* are eligible to be added to the MIS; a node $v \in c$ is added to the MIS **iff** no one of its neighbors is a member of the MIS. The fact that nodes are added to the MIS based on their D2-colors rather than on their IDs is the reason for the short running time of this algorithm since a node v does not have to wait for all its neighbors with smaller IDs to make their decision before it can decide for its own. In the third phase, every node $v \in MIS$ selects a path, p, to any member $u \in N_3(v)$ **iff** $id(u) > id(v)$. The intermediate nodes of each path p along with the members of the MIS constitute a CDS. The second algorithm of [75] is similar to the first one in that first a MIS set is created then it is connected into CDS. However, in this algorithm there is no D2-coloring phase. To avoid collisions between messages broadcasted during the MIS and CDS creation phases, the collision-free broadcasting algorithm presented in [46] is used. In this algorithm, every node is assumed to know its *2-hop* topology, i.e., the nodes in its *2-hop* neighborhood and the edges between these nodes. During the MIS creation phase, any node, v, can be in one of three possible states (*in, out, unsure*); v is in the *in* state if it is a member of the MIS, the *out* state if one of its neighbors is a member of the MIS; otherwise, it is in the *unsure* state. Initially all nodes are in the *unsure* state and the MIS is empty; subsequently, nodes are added to the MIS in rounds that show high degree of parallelism. In every round only *unsure* nodes can compete to join the MIS. In any round, i, any *unsure* node, v, decides to compete for joining the MIS with a probability $p = \frac{1}{2(N_i(v)+1)}$, where $N_i(v)$ is the number of *unsure* neighbors of v in round i. Once v decided to take place in the competition, it informs its neighbors about its decision. If v experiences a collision or receives a message form other neighbor who intends to join the MIS, v stays in the *unsure* state and repeat its attempt in the next round; otherwise, it changes its state to the *in* state and all its neighbors change their states to the *out* state. This phase ends when every node is either in the *in* state or the *out* state. The CDS creation phase is exactly the same as that of the first algorithm. Butenk *et al.* presented a heuristic

[5]D2-coloring is the process in which every node v is assigned a color c that is different from the color assigned to any node in its *2-hop* neighborhood

algorithm that has a constant approximation ratio of 8 [19]. This algorithm has two phases: a MIS set is created in the first phase and its connected in the second phase. It is assumed that a *Leader* node ϑ already exists in the MANET (otherwise, it has to be elected). Initially, all nodes are *White* colored. The *Leader* node ϑ starts the MIS creation phase by changing its color to *Black* and broadcasting a *Dominator* message. Any *White* node that receives a *Dominator* message, changes its color to *Gray* and broadcasts a *Dominatee* message. Any *White* node γ that receives a *Dominatee* message becomes an *active* node and broadcasts a *Degree* message. This message contains γ's *id* along with the number of γ's white neighbors (d^*). The active node that has the largest (d^*, id) changes its color to *Black*. The *Gray* node that has no *White* neighbors sends a *NumberOfBlackNeighbors* message to report the number of its black neighbors. When the *Leader* node ϑ receives a *NumberOfBlackNeighbors* message from all its neighbors; ϑ ends the first phase. In the second phase, the *Black* node that is not *dominated* by any other node is called *active* node, initially all black nodes are active. The *Gray* node that has at least one active neighbor is called an *effective* node. The *Leader* node ϑ starts the second phase by sending an *exploration* message (M) to its *Gray* neighbor γ that has the largest number of black neighbors. Upon the reception of M, γ changes its color to *Black*, broadcasts a *Parent* message declaring ϑ as its *Dominator*, and sends *exploration* message, (M), to any one of its active neighbors. When an active node ω receives an *exploration* message, M, from a *Gray* node γ, it sets its *Dominator* to γ, broadcasts a *Parent* message, and sends *exploration* message to the *Gray* neighbor that has the largest number of black neighbors. Once the *Dominator* of a *Black* node is set, that node is considered *inactive*. Upon the reception of a *Parent* message, a *Gray* node υ broadcasts a *NumberOfBlackNeighbors* message to report the number of its active neighbors, if this number is zero, υ is considered *ineffective*. Any *effective* node that has no *active* neighbors (or *active* node that has no *effective* neighbors) sends *Done* message to its *Dominator*. The second phase ends when the *Leader* node receives *Done* messages from all its neighbors. The CDS consists of all black nodes. A very similar algorithm is presented by Cheng and Du in [29]. In [20], Butenk *et al.* presented a heuristic algorithm that creates a feasible CDS at any time of its execution. Two versions of this algorithm are presented: a *centralized* version and a *distributed* one. Both versions of the algorithm start by including all nodes of the MANET in the CDS set D. All members of the initial CDS are called *Non-fixed* CDS members. The *centralized* version of the algorithm proceeds in the following iterative manner:

1. Select the node $\upsilon \in D$ that has the lowest degree in the subgraph induced by

the D, $G[D]$.

2. Remove υ form D **iff** the subgraph induced by $D - \upsilon$ is connected. Otherwise, υ is considered a *fixed* CDS member.

3. If υ is removed from D and υ has no neighbor that is a fixed member of D, select υ's *Non-fixed* neighbor that has highest degree and fix it.

In the *distributed* version, it is assumed that a node, γ, that has the smallest degree among all nodes is elected as a *Leader*. At any time, only one node is selected to run the algorithm. at the beginning, the leader node γ is selected to start the algorithm. The selected node, ϑ, tests the connectivity of $D - \vartheta$. If removing ϑ disconnects D, ϑ fixes itself as a new dominator by sending a *NEWDOM* message to its neighbors and selects its lowest degree neighbor to run the algorithm by sending *TRY-DISCONNECT* message to this neighbor. Otherwise, ϑ removes itself from D by sending *DISCONNECTED* message to its neighbors; if none of ϑ's neighbors is fixed, ϑ fixes its highest degree neighbor, υ, by sending *SET-DOMINATOR* message to υ. After receiving the *SET-DOMINATOR* message, υ starts running the algorithm.

3.1.4 Clustering using weakly connected dominating sets

The use of weakly connected dominating sets (WCDS) for clustering MANETs was introduced by Chen and Liestman in [28]. In this work, Chen and Liestman proposed two centralized algorithms along with their distributed versions. Both of these algorithms are based on the second algorithm of Guha and Khuller [50]. In the first algorithm, every node has one of three possible colors: *Black*, *Gray*, and *White*. Initially, all nodes are White colored. The term *White Piece* refers to a *White* node, and the term *Black piece* refers to the maximum set of *Black* nodes that induce a weakly connected subgraph and the *Gray* nodes that have at least one neighbor from these *Black* nodes. The algorithm proceeds in iterative manner. In every iteration the *White* or *Gray* node that achieves the maximum reduction in the number of *White Pieces* is colored *Black* while all its *White* neighbors are colored *Gray*. The algorithm ends when all nodes are colored either *Black* or *Gray*. The distributed version of this algorithm starts by building a spanning tree for the MANET. Once the tree is built, every iteration the root of the tree sends a query asking for the node that can achieve the maximum reduction in the number of pieces. Once the optimal node is determined, it is added to the WCDS and a new iteration is initiated. The algorithm ends when all nodes of the MANET are either *Black* or *Gray*. In the second algorithm, initially all nodes are *White* colored and the WCDS is empty. The algorithm proceeds in iterations. Every iteration a *White* or *Gray* node, γ, is colored *Black* and all its neighbors are colored *Gray*; then γ and its neighbors join the WCDS. In the first iteration, γ is selected randomly and in the subsequent iterations any *White* node that has at least one neighbor in the WCDS or any *Gray* node is a candidate. The candidate that covers the maximum number of *White* nodes is selected. The algorithm ends when all nodes are either *Gray* or *Black*. In the distributed implementation of this algorithm, the spanning tree is constructed iterativly. The first node added to the WCDS functions as the root of the spanning tree and all its neighbors function as tree leafs. In every iteration, the root node sends a query to the tree leafs looking for the optimal node among the leafs and their *White* neighbors. Once the optimal node is determined, it is added to the WCDS and all its neighbors are added to the tree leafs. The algorithm ends when there is no *White* node in the MANET. It is apparent that these two algorithms are not localized since they need global knowledge of the improvement achieved by every candidate node in order to decide which node to be added in every iteration. An algorithm that is more localized than all the algorithms presented in [28] is introduced in [25]. In this algorithm, the MANET is partitioned into a set of non-overlapping zones. To partition the MANET, a spanning forest is constructed in the first phase. Every tree of the spanning forest represents a zone. A WCDS is constructed in every zone using the distributed version of the first algorithm presented in [28]. The WCDS of every zone is connected to the WCDS sets of the neighboring zones using boundary nodes, i.e., nodes that have contact with members of more than on zone. In [26], Chen and Liestman addressed the maintenance of the WCDS constructed by the their algorithm presented in [25].

The fact that *the MIS set S in which the shortest bath between any subset V and its complement set $MIS - V$ is exactly 2 hops is a WCDS that has an approximation factor of 5* is proved by Alzoubi et. al in [8]. Based on this fact, Alzoubi et. al proposed the use of their MIS creation algorithm presented in [7] to construct a WCDS [6]. although this algorithm has good approximation factor, the fact that it starts by creating a spanning tree makes it inapplicable in highly mobile environments. Another algorithm that is more suitable to MANETs is presented in [8]. This algorithm has an approximation factor of 122.5. Even though the previous algorithm has better approximation ratio, this algorithm is better in terms of messaging overhead [8]. In the first phase of this algorithm a MIS set is created

[6]The details of this algorithm were explained in the previous section

using the same algorithm employed in [6] [7]. At the end of the first phase, the WCDS consists of all members of the MIS. In the second phase, every pair of MIS members that are 3 hops apart select one intermediate node and add it to the WCDS.

3.1.5 Clustering using other heuristics

Clusters that have no clusterheads are proposed by Vaidya et al. In [88]. In this algorithm every group of nodes that induces a complete graph, known as *clique*, represents a cluster. Nodes that belong to more than one clique are known as *boundary nodes* and they are used to relay messages between clusters. McDonald and Znati proposed a clustering scheme in which path availability is employed to build dynamic clusters [68]. In this algorithm, a (α, t)-Path is a chain of links that is available for time $\geq t$ with a probability $\geq \alpha$. Nodes that can reach each other over (α, t)-Paths form a (α, t)-Cluster. Mcdonald and Znati proposed a set of path selection techniques in [70, 71]; moreover, they presented a model for path stability in [69].

[7]The details of this MIS creation algorithm were given in the previous section

Chapter 4

Details of the Studied Algorithms

4.1 Introduction

This chapter gives a detailed description of the CDS-based clustering algorithms to be studied experimentally in this thesis, namely Wu and Li algorithm, Stojmenovic algorithm, the MPR-CDS algorithm, and Alzoubi algorithm.

4.1.1 The rationale behind our selection

A question arises naturally at this point pertaining to the rationale behind choosing these specific algorithms in this comparative study. The answer to this question is the following: due to their short running time, light computational burden, and low messaging overhead, Wu and Li algorithm, its variant presented by Stojmenovic, and the MPR-CDS algorithm are attractive candidates for clustering MANETs. No simulation-based comparison of their performance in clustering MANETs is available in the literature. To the best of our knowledge, the only simulation-based comparison of the performance of these algorithms was conducted in the context of clustering sensor networks [15]. The maintenance costs incurred by each of these algorithms represent an important factor that has not been evaluated and has always been overlooked. As a matter of fact, some researchers have anticipated that the messaging overhead incurred by Wu and Li algorithm in maintaining the CDS might be so heavy to the extent that makes Wu and Li algorithm an infeasible solution for MANETs clustering; however, this is not supported by any simulation-based evidence [96]. The fourth algorithm investigated in this thesis is that presented by Alzoubi et al. in [6]. The designers of this algorithm claim that it outperforms

Algorithm	Wu and Li	Stojmenovic	Alzoubi et al.
Approximation Factor	$\frac{n}{2}$	n, $\frac{n}{2}$	192
Message Complexity	$\Theta(m)$	$O(n)$	$O(n)$
Time Complexity	$O(\Delta^3)$	$\Omega(n)$	$O(n)$

Table 4.1: Message and time complexity of Wu and Li , Stojmenovic, and Alzoubi algorithms.

Wu and Li algorithm and its Stojmenovic variant in terms of speed and messaging overhead. A theoretical comparison among the performance metrics of Wu and Li algorithm, its variant presented by Stojmenovic, and Alzoubi algorithm is given in Table 4.1. As proven in [6], the time complexity of Alzoubi algorithm is $O(n)$ where n is the number of mobile nodes whereas the time complexity of Wu and Li algorithm is shown to be $O(\Delta^3)$ where Δ is the maximum node degree. The message complexity of Alzoubi algorithm is shown to be $O(n)$ while that of Wu and Li algorithm is $\Theta(m)$ where m is the number of edges in the network. In dense networks, Δ can be equal to n, and m can be equal to n^2, i.e., the time and message complexities of Wu and Li algorithm in dense networks can be equal to $O(n^3)$ and $\Theta(n^2)$, respectivly [89]. The design and the performance analysis of these algorithms is based on the assumption that all packets broadcasted by a given node are successfully received by all its neighbors. The fact that this assumption is far away from reality, since broadcasting in MANETs leads to loss of packets due to collisions, imposes thick shadows of doubt about the significance of theoretical performance analysis, moreover, it emphasizes the importance of simulation-based performance analysis.

4.1.2 The aspects of comparison

The time spent, the messaging overhead generated in CDS creation and maintenance phases, and the size of the resulting CDS are the comparison aspects that are used in this thesis. Moreover, the effect of MANET size, node mobility model, and node speed on these metrics is investigated.

4.2 Details of the algorithms

This section gives a detailed description of the CDS creation and maintenance phases of each algorithm.

4.2.1 Wu and Li clustering algorithm

Wu and Li [92] clustering algorithm is a simple, fast and localized algorithm in which every node ϑ uses the knowledge of its *2-hop* neighborhood to determine whether it belongs to the CDS or not. The algorithm is implemented as follows:

1. Initially all nodes are marked *white*, i.e. the CDS is empty.
2. Every node exchanges information about its *1-hop* neighborhood with its neighbors.
3. After receiving this information from all of its neighbors, node ϑ decides about its *tentative color*. The *tentative color* of ϑ is *black* and ϑ is a CDS member if it has two neighbors u and w such that u and w are not neighbors of each other, (CDS members are called *gateways* in [92]). Otherwise, ϑ's tentative color is *white* and it is a regular node. ϑ announces its *tentative color* to its neighbors.
4. After receiving the *tentative colors* of all its neighbors, A *black* node ϑ withdraws form the CDS by remarking itself *white*, and announces its final color to its neighbors if any of the following two conditions is satisfied:
   - ϑ has a neighbor u such that N[v]$\subseteq$ N[u] and *id(v)* $<$ *id(u)*, where N[v] is the closed *1-hop* neighborhood of node v.
   - ϑ has two neighbors u and w such that u and w are neighbors of each other and N[v]$\subseteq$ N[u]$\cup$ N[w] and *id(v)*= $min(id(v), id(u), id(w))$.

Step 4 of this algorithm is called the pruning step and its objective is to reduce the size of the resulting CDS by pruning the redundant gateways. Examples of applying these simple rules to create a CDS and to reduce its size are given in Figure 4.1 and Figure 4.2 respectively. In [36], Dei and Wu generalized the pruning rule as follows: *Let $C = \{u_1, u_2, ..., u_k\}$ be a set of gateways that induces a complete graph G[C]; a gateway v is pruned* ***iff*** $N(v) \subseteq \bigcup_{u \in C} N(u)$ *and* $id(v) < min(id(u) : u \in C)$.

The maintenance of the CDS in face of node mobility is achieved as follows:

1. A moving node ϑ exchanges information about its *1-hop* neighborhood with its neighbors every τ time units.

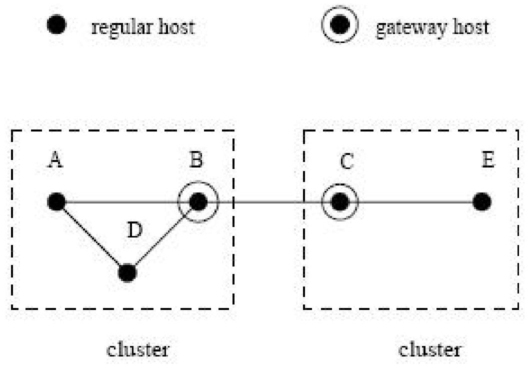

Figure 4.1: CDS created using Wu and Li Algorithm.

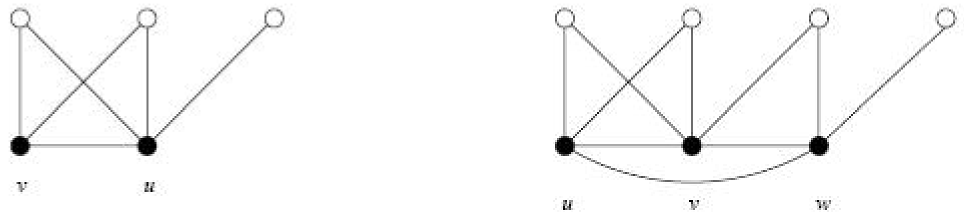

Figure 4.2: Reducing the CDS size.

2. Every node $u \in \vartheta \bigcup N(\vartheta)$ updates its role according to the connectivity of its current *2-hop* neighborhood by applying steps 2, 3, and 4 of the original algorithm.

3. A gateway node w that has a broken link to ϑ exchanges information about its *1-hop* neighborhood with its neighbors, w withdraws from the CDS if all the members of its current *1-hop* neighborhood are neighbors of each other.

4.2.2 Stojmenovic clustering algorithm

A modified version of Wu and Li algorithm that has a better approximation factor and lower messaging overhead is presented by Stojmenovic et al. in [85]. This version of the algorithm works as follows:

1. Initially all nodes are marked *white*, i.e. the CDS is empty.

2. Every node, ϑ, informs its neighbors about its *1-hop* neighborhood and about its weight expressed as $< \vartheta's\ degree, \vartheta's\ ID >$.

3. After receiving the weight and neighborhood information from all of its neighbors, node ϑ ranks its neighbors according to their weights in lexicographic order.

4. Node ϑ marks itself *black* and becomes a member of the CDS if at least two of its neighbors are not adjacent to each other and its neighborhood is not covered by a higher rank neighbor or by a connected set of higher rank neighbors.

5. Every node informs its neighbors about its last decision regarding the CDS membership.

Notice that Stojmenovic version of Wu and Li algorithm spares nodes the transmission of the *tentative color* messages. Stojmenovic algorithm use the maintenance procedure of Wu and Li algorithm.

4.2.3 The MPR clustering algorithm

In [80], Qayyum et al. used the concept of *Multi point Relays* to design an efficient broadcasting algorithm that reduces the number of transmitting nodes while guaranteeing the delivery of broadcast messages to all network nodes. In this algorithm, every node v uses the knowledge of its *2-hop* neighborhood to elect a set S of its immediate neighbors to cover all of its *2-hop* neighbors. The members of S are called the *Multi point Relays*, *MPR*, of v. It is obvious that the set S produced by this algorithm is *source-dependent MPR*. A node v constructs its *MPR* set as follows:

- *step 1*: Node $u \in N_1(\text{v})$ is added to S **iff** u is the only *1-hop* neighbor of v that covers some of v's *2-hop* neighbors.

- *step 2*: Node $w \in N_1(\text{v})$ is added to S **iff** w covers the largest number of yet uncovered *2-hop* neighbors of v. This step is repeated until all members of $N_2(\text{v})$ are covered.

- *step 3*: Node v informs all members of S that they have been added to v's *MPR*.

Adjih et al. [2] modified the work in [80] in such a way that the new algorithm produces *source-independent MPR* called the *MPR-CDS*. The algorithm starts by having every node v calculate its *source-dependent MPR* as in [80]; after that, every node v decides whether it belongs to the *MPR-CDS* or not according to the following simple rules:

- *Rule 1*: Node $v \in$ *MPR-CDS* **iff** v has the smallest *ID* in its *1-hop* neighborhood.
- *Rule 2*: Node $v \in$ *MPR-CDS* **iff** $v \in w$'s *MPR* where w's ID is the smallest in v's *1-hop* neighborhood.

The proof of correctness of this simple localized algorithm is given in [2]. In [91], Wu has noticed that in many occasions nodes added by *Rule 1* of the *MPR-CDS* algorithm are *useless*; moreover, the algorithm used to calculate the *source-dependent MPR* does not benefit from *Rule 2* of the *MPR-CDS* algorithm. In [91] Wu modified *Rule 1* as follows:

Node $v \in$ MPR-CDS ***iff*** *v has the smallest ID in its 1-hop neighborhood* ***and*** *v has at least two unconnected neighbors.*

Moreover, Wu modified the *MPR* calculation algorithm introduced in [80] by having every node v start by adding all its free neighbors to its *MPR* set. A node u is a *free neighbor* of node v *iff* $u \in N_1(v)$ **and** v is not the smallest ID neighbor of u. As reported by Wu [91], these modifications reduce the size of the resulting *MPR-CDS* and hence improve its approximation factor. We will use the name *extended MPR-CDS* to refer to this algorithm. According to [15] the *extended MPR-CDS* outperforms Wu and Li algorithm in terms of speed and messaging overhead.

The designers of this algorithm do not provide any maintenance mechanism to deal with topology changes that affect the domination and/or the connectivity properties of the CDS. However, the maintenance approach used in this study is as follows:

- Every node ϑ keeps track of and informs its neighbors about changes in its 1-hop neighborhood.
- Node ϑ invokes the CDS creation algorithm in any of the following cases:
    - A change in ϑ's 1-hop neighborhood leads to a change in its 2-hop neighborhood.
    - ϑ losses a member of its MPR set.

4.2.4 Alzoubi clustering algorithm

In [6], Alzoubi et al. proposed a distributed approximation algorithm for the construction of a connected dominating set in unit disk graphs. The designers of this algorithm claim that their algorithm is the only one that has a constant approximation factor (it is equal to 192), preserves its approximation factor after maintenance operations, and is faster and generates less overhead in dense networks than Wu and Li algorithm and its Stojmenovic variant. A detailed description of the CDS construction and maintenance phases is given in the sequel.

The CDS construction phase

This phase consists of two stages. In the first stage a maximal independent set MIS is created, then nodes are added to connect the MIS in the second stage. The MIS is created as follows:

- Initially all nodes are colored *White*.
- A node v declares itself a *Dominator* and changes its color to *Black* **iff** it has the smallest ID among all its neighbors. All neighbors of v change their colors to *Gray* and declare themselves *Dominatee* nodes.
- If all neighbors with smaller IDs of a white node u are *Dominatees*, node u declares itself a *Dominator* and changes its color to *Black*

The MIS creation stage terminates when all nodes are colored either *Gray* or *Black*. The MIS is composed of all the *Black* nodes. It is proven in [7, 4] that the MIS created by this algorithm satisfies the following property:

Any set $U \subset MIS$ is at most 3-hops away from its complement $V = MIS - U$.

In the second stage, every Dominator v selects a path P to every *2-hop* and *3-hop Dominator* neighbor that has a larger ID than that of v. The intermediate nodes of each path P, at most 2 *Dominatees*, are added to the set of *Connectors* C. The CDS is composed of all nodes $u \in MIS \cup C$.

Implementation details

Any node V can be in one of four possible states: *Candidate*, *Dominator*, *Dominatee*, or *Connector*; node V enter the *Connector* state only from the *Dominatee* state.

A. Local Variables and Data Structures

This section gives a description of the variables and data structures maintained in every node depending on its state.

1. Every node V, regardless of its state, maintains the following variables and data structures:
   - A list, *nlist*, of which every entry contains the ID of a neighbor node along with its state.
   - A counter, x_1, that stores the number of *Candidate* neighbors.
   - A counter, x_2, that stores the number of neighbors that have smaller IDs.
2. Every *Dominatee* node, E, maintains the following additional variables and data structures:
   - A list, $list_1$, of which every entry stores the ID of a *Dominator* neighbor.
   - A list, $list_2$, of which every entry stores the ID of a 2-hop *Dominator*, υ, along with the IDs of the *Dominatees* that are common neighbors of E and υ; these *Dominatees* neighbors are called the *reporting neighbors set*, *rns*, of υ.
   - A counter, y, that stores the number of *Dominatee* neighbors that have reported their $list_1$ and $list_2$ lists.
3. Every *Dominator* node, R, maintains the following additional variables and data structures:
   - A counter, z, that counts the number of *Dominatee* neighbors that have not reported their $list_1$ and $list_2$ lists yet.
   - A list, $2Hlist$, of which every entry contains the ID of a 2-hop *Dominator* neighbor, υ, along with the IDs of υ's *rns*.
   - A list, $3Hlist$, of which every entry contains the ID of a *Dominator* neighbor, ν, whose ID is larger than that of R and that can be reached only via 3-hop paths along with the IDs of R neighbors these paths; these neighbors are called υ's *rns*.

- A list, $3Hlist2$, of which every entry contains the ID of a *Dominator*, v, whose ID is larger than that of R and that can be reached via 2-hop and 3-hop paths along with the IDs of R's neighbors that offer the 3-hop paths to v, these neighbors are called v's rns.
- A list, *SClist*, that keeps information about the *Connector* neighbors that R uses to reach other *Dominators*. Every entry in this list contains the following:
    - The ID of a selected connector k.
    - The set of 2-hop *Dominator* neighbors that can be reached through k; these *Dominators* are known as the *2-hop* target set $ts2$.
    - The set of 3-hop *Dominators* that can be reached via k; these *Dominators* are known as the *3-hop* target set $ts3$.
- A list, *AClist*, that keeps information about all *Connector* neighbors. Every entry in this list contains the following:
    - The ID of a *Connector* neighbor k.
    - The IDs of the 2-hop *Dominators* that are adjacent to k.
    - The IDs of the 3-hop *Dominators* that can be reached via k.

4. In addition to the variables and data structures maintained by *Dominatee* nodes, every *Connector* node, C, maintains the following variables and data structures:

    - A list, $Rlist_1$, of which every entry stores the ID of an adjacent *Dominator* u that uses C to connect to other *Dominators* that are 1-hop away from C along with the IDs of these *Dominators*; the set of these *Dominators* is known as the $1hds$ set.
    - A list, $Rlist_2$, of which every entry stores the ID of an adjacent *Dominator* u that uses C to connect to other *Dominators* that are 2-hop away from C along with the IDs of these *Dominators*; the set of these *Dominators* is known as the $2hds$ set.
    - A list, $Aclist_1$, of which every entry stores the ID of an adjacent *Connector* w, the set of *Dominators* that are adjacent to w and that use w as a connector, this set is called the set of selectors ss, and the set of *Dominators* that are adjacent to C and are connected to the members of w'ss through the pair (w, C); the later set is called the *associated target dominators set atds*.

- A list, *Aclist*$_2$, of which every entry stores the ID of an adjacent *Connector* w, the set of *Dominators* that are adjacent to C and use it as a connector ss, and the set of *Dominators* that are adjacent to w and are connected to the members of $C'ss$ through the pair (C, w), this later set is w's $atds$.

B. Messages and Actions

Initially all nodes are in the candidate state and are colored *White*; nodes change their colors and states and take actions based on their current states and on the type of messages received from their neighbors. The rules that govern node behavior are as follows:

- A candidate node ν that has the lowest ID among all its neighbors changes its state to *Dominator* state, marks itself *Black*, and broadcasts a *Dominator* message. Moreover, ν sets the value of counter z to twice the number of its neighbors.
- Upon the reception of a *Dominator* message, a candidate node υ updates the sender's state in its *nlist*, inserts the sender's ID in its $list_1$, changes its state to *Dominatee* state, marks itself *Gray*, broadcasts a *Dominatee* message, decrements the value of counter x_1 by one, and if the new value of x_1 is zero, υ broadcasts a LST_1 message that contains υ's ID and its $list_1$ list.
- Upon the reception of a *Dominator* message, a *Dominatee* node ω updates the sender's state in its *nlist*, inserts the sender's ID in the $list_1$, decrements the value of counter x_1 by one, and if the new value of x_1 is zero, ω broadcasts a LST_1 message that contains $\omega's$ ID and its $list_1$ list.
- Upon the reception of a *Dominatee* message, a candidate node υ updates the sender's state in its *nlist*, decrements the value of its counter x_1 by one; if the sender's ID is lower than its own, υ decrements the value of its counter x_2 by one, if the new value of x_2 is zero, υ changes its state to the *Dominator* state, marks itself *Black*, sets the value of counter z to twice the number of its neighbors, and broadcasts a *Dominator* message.
- Upon the reception of a *Dominatee* message, a *Dominatee* node ω updates the sender's state in its *nlist*, decrements the value of its counter x_1 by one; if the new value of x_1 is zero, ω broadcasts a LST_1 message that contains ω's ID and its $list_1$ list.

- Upon the reception of a LST_1 message, a *Dominator* node υ decrements the value of its counter z by one, for every entry of LST_1 that contains a *Dominator* ω whose ID is larger than that of υ and is not included in υ's *2Hlist*, υ inserts ω in its *2Hlist* and adds the sender's ID to the *rns* associated with ω, if ω is already included in the *2Hlist*, υ just adds the sender's ID to ω's *rns*.
- Upon the reception of a LST_1 message by a *Dominatee* node ν, for each *Dominator* υ that is included in the LST_1 message and not included in ν's $list_2$, ν adds an entry to its $list_2$ in which it stores υ's ID and adds the sender's ID to υ's *rns*; for all the *Dominators* that are already included in ν's $list_2$, ν adds the sender's ID to the (*rns*) associated with each *Dominator*. Moreover, ν increments the value of its counter y by one, if the value of x_1 is equal to zero and the new value of y is equal to the number of *Dominatee* neighbors, ν broadcasts a LST_2 message. This message is composed of ν's ID and the IDs of all *Dominators* in ν's $list_2$ list.
- Upon the reception of a LST_2 message by a *Dominator* node υ, for each *Dominator* ω that is included in LST_2 message and not included in υ's *2Hlist*, *3Hlist*, and *3Hlist2* lists and whose ID is larger than υ's ID, υ adds an entry to its *3Hlist* in which it ω's ID and adds the sender's ID to the *rns* associated with ω; if ω is already included in υ's *2Hlist*, υ adds an entry to its *3Hlist2* and in which it inserts ω's ID and adds the sender's ID to the *rns* associated with ω; if ω is already included either in υ's *3Hlist* or its *3Hlist2*, υ just adds the sender's ID to the *rns* associated with ω in that list. Moreover, υ decrements the value of counter z by one; if the new value of z is zero, υ proceeds as follows:
  - Based on the information included in its *2Hlist* and *3Hlist* lists, υ selects some of its *Dominatee* neighbors to connect itself to all *Dominators* included in its *2Hlist* and *3Hlist* lists.
  - For every selected *Dominatee*, ω, an entry is added to υ's *SClist*; this entry contains ω's ID along with the IDs of the set of *Dominators*, Γ, that are reached by υ through ω. The same entry is added to υ's *AClist* if ω is not included in this list; if it is already included, υ updates the set of *Dominators* that can be reached through ω to include the members of Γ.
  - υ unicasts a *Select1* message to every *Dominatee* neighbor it has selected as a *Connector* in the previous step. This message contains

υ's ID and a copy of its *SClist*.

- If υ has the smallest ID among all its *2-hop* and *3-hop Dominator* neighbors, υ broadcasts a *Complete* message.

- Whenever a *Dominatee* node receives a *Select1* message addressed to itself, it changes its state to *Connector*, marks itself *Black* and inserts a new entry in its $Rlist_1$ (respectively, $Rlist_2$) in which it stores the sender's ID along with the $ts2$ (respectively, $ts3$) set associated with its ID in the *SClist* of the received *Select1* message.

- Once it receives *Select1* messages, not necessarily addressed to itself, from all its *Dominator* neighbors except the one with the largest ID, a *Connector* node ν whose $Rlist_2$ is not empty, chooses a set, Γ, of its *Dominatee* and *Connector* neighbors that cover all members of the $2hds$ sets listed in its $Rlist_2$. For every $\upsilon \in \Gamma$, ν inserts an entry in its $Aclist_2$ in which it stores υ's ID, the *selectors set*, ss, that ν connects to other *Dominators* through υ, and the set of *Dominators* that can be reached by ss through the pair of connectors (ν, υ) ; these *Dominators* are known as the *associated target dominators set* ($atds$). After that, ν unicasts a *Connect1* message that contains ν's ID, a copy of ν's $Rlist_1$, and a copy of its $Aclist_2$ to every *Dominators* included in the $1hds$ sets of ν's $Rlist_1$, and every *Dominatees* included in its $Aclist_2$.

- Whenever a *Dominator* ϑ receives a *Connect1* message addressed to itself from a neighboring *Connector* ω, it inserts a new entry in its *AClist* in which it stores the ID of ω and adds to the $ts2$ associated with this entry the ID of every *Dominator* υ that is included in the $Rlist_1$ of the received message and that uses ω to connect to ϑ. However, If ω is already included in ϑ's *AClist*, ϑ just adds the ID of υ to the $ts2$ associated with ω.

- Upon the reception of a *Connect1* message addressed to itself, a *Dominatee* or a *Connector* ν proceeds as follows:

  - If ν is a *Dominatee*, it changes its state to *Connector* and marks itself *Black*.
  - Node ν inserts a new entry in its $Aclist_1$ in which it stores the sender's ID along with the ss and $atds$ sets associated ν's ID in the $Aclis_2$ of the received *Connect1* message.
  - A new entry is inserted in ν's $Rlist_2$ for every *Dominator* ω included in the $atds$ set associated with ν's ID in the $Aclis_2$ of the received

Connect1 message; this entry stores ω's ID and the $2hds$ of this entry is set to the ss set associated with ν's ID in the $Aclis_2$ of the received *Connect1* message. If ω's ID is already included in ν's $Rlist_2$, the ss set is added to the $2hds$ associated with ω.

- Node ν unicasts a *Connect2* message to every *Dominator* included in the $atds$ set associated with ν's ID in the $Aclis_2$ of the received *Connect1* message. This message is composed of ν's ID, the ss and $atds$ sets associated with ν's ID in the $Aclis_2$ of the received *Connect-1* message.

- Whenever a *Dominator* node υ receives a *Connect2* message addressed to itself from a *Connector* neighbor ω, υ inserts a new entry in its *AClist* in which it stores ω's ID and it sets the $3hds$ of this entry to the ss set of the received *Connect2* message; if ω is already included in υ's *AClist*, the ss set is added to the $3hds$ associated with ω.
- Whenever a *Dominator* node υ has selected *Connectors* to all *Dominators* that have larger IDs and are at most 3-hop away and it is reached by all *Dominators* that have smaller IDs and are at most 3-hop away, ϑ broadcasts a *Complete* message.

The CDS maintenance phase

In this algorithm, the approach used to maintain the CDS after any topological change that renders the current CDS invalid is to first maintain the MIS and then to make sure that all pairs of MIS members that are *3-hop* away (at most) are connected through *Connector* nodes. A detailed description of the maintenance procedures of Alzoubi algorithm is given below.

Maintenance Due to Dominator Movement

- Whenever a *Dominatee* node ϑ discovers the disappearance of a *Dominator* neighbor ω it proceeds as follows:
  - ϑ removes ω from its $list_1$ and $nlist$ lists.
  - If ϑ's $list_1$ becomes empty, it changes its state to *Candidate* and clears all variables associated with its previous state. Otherwise, ϑ keeps its *Dominatee* state.

  - ϑ reports the loss of ω by broadcasting a *Warning1* message that contains its own ID, its current state, and the ID of ω.

- Whenever a *Connector* node ϑ discovers the disappearance of a *Dominator* neighbor ω, it proceeds as follows:

  - ϑ removes ω from its $list_1$, and *nlist* lists. If $list_1$ becomes empty, ϑ changes its state to *Candidate* and clears all variables associated with its previous state.
  - If ϑ has other *Dominators*, i.e., its $list_1$ is not empty, it removes ω from the $1hds$ of each entry in its $Rlist_1$; if the $1hds$ of that entry becomes empty, the whole entry will be removed form $Rlist_1$. Moreover, if ω appears as the first parameter of any entry in ϑ's $Rlist_1$ or $Rlist_2$, i.e., ω is a *Selector* of ϑ, that entry is removed. If both of $Rlist_1$ and $Rlist_2$ become empty, ϑ changes its state to *Dominatee* and clears all variables associated with its previous state.
  - If ϑ's $Rlist_1$ or $Rlist_2$ or both are not empty, i.e., ϑ is still selected as *Connector*, it removes ω form the *atds* (respectively, *ss*) of every entry in its $Aclist_1$ (respectively, $Aclist_2$); if that *atds* (*ss*) becomes empty, the whole entry will be removed.
  - ϑ reports the loss of ω by broadcasting a *Warning1* message that contains its own ID, its current state, and the ID of ω.

- Upon the reception of a *Warning1* message form a neighbor ϑ reporting that ϑ changed its state from a *Connector* to a *Dominatee* or to a *Candidate* due to the loss of connection with a *Dominator* neighbor ω, a *Connector* node ν proceeds as follows:

  - If ϑ is included in ν's $Aclist_1$, the entry corresponding to ϑ is removed.
  - If ϑ is listed in ν's $Aclist_2$ and ω is listed in the *atds* associated with ϑ in this list, and if ω is still listed in ν's $list_2$, ν selects a new node from the *rns* associated with ω in its $list_2$ to connect itself to ω and applies the CDS algorithm locally starting at the MIS connecting phase. Moreover, ν removes ϑ from its $Aclist_2$.

- Upon the reception of a *Warning1* message form a *Connector* ϑ reporting the loss of connection with a neighboring *Dominator* ω, a *Connector* node ν proceeds as follows:

- If ϑ is listed in ν's *Aclist*$_1$ list and ω belongs to the ss associated with ϑ in this list, ν removes ω from ss and if it becomes empty, ν removes the entire entry of ϑ from its *Aclist*$_1$.
- If ϑ is listed in ν's *Aclist*$_2$ and ω belongs to the *atds* associated with ϑ in this list, ν removes ω from the *atds*, and if it becomes empty, ν removes the whole entry of ϑ from its *Aclist*$_2$. Moreover, ν removes ϑ form the rns associated with ω in its $list_2$, and if this rns becomes empty, ν removes ω from its $list_2$ and from the $2hds$ of each entry of its *Rlist*$_2$ and if any $2hds$ becomes empty, the whole entry will be removed from ν's *Rlist*$_2$; if ν's *Rlist*$_2$ becomes empty and its *Rlist*$_1$ is empty, ν changes its state to *Dominatee*. However, if the rns associated with ω in ν's $list_2$ is not empty after the removal of ϑ, ν selects a node from the rns associated with ω in its $list_2$ to connect itself to ω, and applies the CDS algorithm locally starting at the MIS connecting phase.
- Regardless of whether ν has changed its state or not, if ω is removed from ν's $list_2$, ν reports the loss of ω by broadcasting a *Warning2* message that is composed of ν's ID, ν's current state, and ω's ID.
- If ω is still listed in ν's $list_2$, and ν has not sent a *Warning1* message, it broadcasts a *Response* message that contains its ID, its current state, and ω's ID.

• Whenever a *Dominator* ϑ receives a *Warning1* message form a neighbor ν reporting the loss of connection with a *Dominator* ω, it proceeds as follows:

- ϑ removes ν from the rns set associated with ω in its *2Hlist*; if the rns becomes empty, ϑ removes the whole entry of ω from its *2Hlist*, and If ω is listed in any entry of ϑ's $3Hlist_2$, this entry will be moved to the *3Hlist*.
- If ν is the *Connector* that is used by ϑ to reach ω, i.e., ω belongs to the $ts2$ ($2hds$) associated with ν in ϑ's *SClist* ($AClist$), ϑ removes ω from this $ts2$ ($2hds$) and if the $ts2$ ($2hds$) becomes empty, and the $ts3$ ($3hds$) of this entry is also empty, the whole entry is removed from ϑ's *SClist* ($AClist$). However, if ω is still listed in ϑ's *2Hlist* or *3Hlist*, ϑ selects a node from the rns associated with ω in its *2Hlist* or *3Hlist* lists and applies the CDS algorithm locally starting at the MIS connecting phase.

• Whenever a *Dominator* ϑ receives a *Warning2* message form a neighbor ν reporting the loss of a *Dominator* ω, it proceeds as follows:

- ϑ removes ν from the *rns* set associated with ω in its *3Hlist*; if the *rns* becomes empty, ϑ removes the whole entry of ω from its *3Hlist.* ϑ applies the same rule to its $3Hlist_2$ if ω is listed in ϑ's $3Hlist_2$.
- If ν is the *Connector* that is used by ϑ to reach ω, i.e., ω belongs to the $ts3$ ($3hds$) associated with ν in ϑ's *SClist* (*AClist*), ϑ removes ω from this $ts3$ ($3hds$) and if the $ts3$ ($3hds$) becomes empty, and the $ts2$ ($2hds$) is also empty, the whole entry of ν is removed from ϑ's *SClist* (*AClist*). However, if ω is still listed in ϑ's *3Hlist*, ϑ selects one of the *rns* nodes associated with ω in its *3Hlist* list and applies the CDS algorithm locally starting at the MIS connecting phase.

- Whenever a *Candidate* node ϑ receives a *Warning1* or *Response* message form each neighbor that is either a *Dominatee* or a *Connector*, it invokes the CDS algorithm starting from the MIS construction phase.

- When a *Dominator* ϑ joins a new neighborhood, it proceeds as follows:

  - If the new neighborhood has at least one *Dominator*, ϑ changes its state to *Dominatee*, updates its variables, and broadcasts a *Dominatee* message, followed by LST_1 message. Whenever a *Dominatee* or a *Connector* node υ receives a *Dominatee* message from the new neighbor ϑ, υ broadcasts a LST_1 message; and when υ receives LST_1 message from the new neighbor ϑ, it updates its $list_2$ and broadcasts LST_2 message. When ϑ receives LST_1 message from each *Dominatee* and *Connector* neighbor, it updates its $list_2$ and broadcasts LST_2 message.
  - On the other hand, if all members of the new neighborhood are in *Non-Dominator* state, ϑ maintains its state as a *Dominator*, updates its variables, and broadcasts a *Dominator* message. Upon the reception of ϑ's *Dominator* message, each member of ϑ's neighborhood adds it to its $list_1$ and then sends LST_1 message followed by LST_2 message. After receiving LST_1 and LST_2 messages from all its neighbors, ϑ updates its variables and executes the CDS algorithm locally starting at the MIS connecting phase.

  Note: In both of the above cases, when a *Dominatee* or a *Connector* node υ receives LST_1 message from a *Dominatee* or a *Connector* neighbor ν that is not a new neighbor, υ recalculates its $list_2$ and broadcasts LST_2 message if the new $list_2$ is different from old one.

Maintenance Due to Dominatee or Candidate node Movement

- Whenever a *Connector* or a *Dominatee* node ω discovers the disappearance of a *Dominatee* neighbor υ, it removes υ from its *nlist* and from the *rns* associated with each entry in its $list_2$. If the *rns* of any entry becomes empty, that entry will be removed from ω's $list_2$ and ω broadcasts a *Lost2* message, which contains its own ID, and the IDs of the *Dominators* removed from its $list_2$; these *Dominators* are called the *Lost-Dominators.* When a *Dominator* ϑ receives the *Lost2* message from ω, it removes ω from the *rns* sets associated with the *Lost-Dominators* listed in its *3Hlist* and *3Hlist2.* If any of the *rns* sets becomes empty, ϑ removes the whole entry associated with that *rns*.

- The actions taken by a *Dominator* ϑ upon the reception of a $Lost2$ message depend on the state of the sender ω; if ω is a *Dominatee*, ϑ removes ω from the *rns* associated each *Lost-Dominator* ν that is listed in its *3Hlist* (respectively, $3Hlist_2$), and if the *rns* becomes empty, the entire entry corresponding to ν is removed. On the other hand, if ω is a *Connector* that is used by ϑ to reach any *3-hop Dominator*, i.e., ω is listed in ϑ's *SClist* (*AClist*), ϑ removes every *Lost-Dominator* ν from the $ts3$ ($3hds$) associated with ω in its *SClist* (*AClist*), if any $ts3$ ($3hds$) set becomes empty, the whole entry is removed. If ν is still listed in ϑ's $3Hlist$, ϑ selects another node among the *rns* associated with ν in its $3Hlist$ to connect itself to ν, and it applies the CDS locally starting at the MIS connecting phase.

- When a *Dominator* node υ discovers the absence of a *Dominatee* neighbor ω, it removes ω from its *nlist*, and from the *rns* set of each entry in its *2Hlist*, *3Hlist* and *3Hlist2.* If any of the *rns* sets becomes empty, υ removes the whole entry of the *Dominator* associated with that empty *rns* set. If ν, the *Dominator* associated with the removed entry, belongs to υ's *2Hlist* and it is still included in υ's *3Hlist2*, υ moves the entire entry of ν to its *3Hlist* list.

- When a *Dominatee* node ϑ joins a new neighborhood, it proceeds as follows:

  - If the new neighborhood has at least one *Dominator*, ϑ updates its *nlist* and broadcasts a *Dominatee* message followed by LST_1 message.
  - If all members of the new neighborhood are in *Non-Dominator* state, ϑ changes its state to *Dominator*, and broadcasts *Dominator* message.
  - In both cases ϑ and its neighbors apply the CDS algorithm locally starting at the MIS connecting phase.

- Whenever a new node υ joins the network, it starts in the *Candidate* state. If any of υ's neighbors is a *Dominator*, υ changes its state to *Dominatee*, and broadcasts a *Dominatee* message followed by LST_1 message; otherwise, υ changes its state to *Dominator*, and broadcasts a *Dominator* message. If the new state of υ is *Dominatee*, υ and its neighbors proceed as follows:

  - Upon the reception of a *Dominatee* message from υ, every *Dominatee* or *Connector* neighbor ω broadcasts LST_1 message.
  - After receiving LST_1 message from υ, each *Non-Dominator* neighbor updates its $list_2$ and broadcasts LST_2 message.
  - Upon receiving LST_1 message from each *Non-Dominator* neighbor, υ updates its $list_2$ and broadcasts LST_2 message.
  - After receiving LST_1 and LST_2 messages from each neighbor, a *Dominator* node ϑ updates its variables.
  - υ and its neighbors apply the CDS algorithm starting at the MIS connecting phase.

  On the other hand, If the new state of υ is *Dominator*, υ and its neighbors proceed as follows:

  - Each *Dominatee* and *connector* neighbor ω adds υ to its $list_1$ and broadcasts LST_1 message followed by LST_2 message.
  - Whenever a *Dominatee* or a *Connector* receives LST_1, it updates its $list_2$ and it broadcasts LST_2 message if the update process has introduced any changes to its $list_2$.
  - After receiving LST_1 and LST_2 messages form all its neighbors, υ and its neighbors apply the CDS algorithm starting at the MIS connecting phase.

Maintenance Due to Connector node Movement

- The actions taken by a node υ due to the absence of a *Connector* neighbor ϑ depend on its state. If υ is a *Dominatee*, υ removes ϑ from its *nlist*, and from the *rns* associated with each *Dominator* in its $list_2$. If the *rns* of any entry becomes empty, υ removes that entry from its $list_2$ and broadcasts a $Lost2$ message that is composed of its ID, and the IDs of the *Dominators* whose entries are removed from its $list_2$, these *Dominators* are called the *Lost-Dominators*. However, If υ is a *Connector*, it takes the additional actions listed below:

- If ϑ uses υ to reach any *2-hop Dominator*, i.e., ϑ is listed in υ's $Aclist_1$, ϑ will be removed from υ's $Aclist_1$.
- If υ uses ϑ to reach any *2-hop Dominator* ω, i.e., there is an entry for ϑ in υ's $Aclist_2$ and ω belongs to the *atds* associated with that entry, and if ω is still listed in υ's $list_2$, υ selects a new neighbor from the *rns* associated with ω to connect itself to ω, and it applies the CDS algorithm locally starting at the MIS connecting phase. On the other hand, if ω is not listed in υ's $list_2$, then υ removes ω from the $2hds$ associated with each entry in its $Rlist_2$, and if that entry becomes empty, then it will be removed. If υ's $Rlist_1$ is empty and its $Rlist_2$ becomes empty, υ changes its state to *Dominatee*, updates its variables, and broadcasts *S-Dominatee* message.

On the other hand, if υ is a *Dominator*, in addition to removing ϑ from its *nlist*, υ takes the following actions:

- υ removes ϑ from the *rns* of each entry listed in its *2Hlist*, *3Hlist* and $3Hlist_2$ lists. If any *rns* set of any entry of these lists becomes empty, the whole entry will be removed. If the removed entry belongs to υ's *2Hlist* and ω is still listed in υ's $3Hlist_2$, then the entry of ω will be moved to υ's *3Hlist*.
- If υ uses ϑ to reach other *Dominators*, i.e., ϑ is listed in υ's *SClist* and *AClist*, υ removes ϑ's entry from these lists, then for each *Dominator* ν that belongs to the $ts2$ ($2hds$) or $ts3$ ($3hds$) of the removed entry and if ν is still listed in any of υ's *2Hlist*, *3Hlist*, or $3Hlist_2$, υ selects a new connector to connect itself to ν by applying the CDS locally starting at the MIS connecting phase.

- Whenever a node ϑ receives a *S-Dominatee* message from a *Connector* neighbor ω, it updates the state of ω in its *nlist* to *Dominatee*. Moreover, if ϑ is a *Dominator* that uses ω to reach other *Dominators*, i.e., ω is listed in ϑ's *SClist* (respectively, *AClist*) list, ω's entry is removed from this list and proceeds as if it lost a *Connector* neighbor.

- Whenever a *Connector* node ϑ moves to a new vicinity, it takes the following actions:

  - If the new neighborhood still contains some *Dominators* that use ϑ to reach each other, ϑ keeps its *Connector* state, updates its variables, and broadcasts a *Connector* message followed by LST_1 message. Whenever a neighboring node receives the LST_1 message, it applies the CDS algorithm locally starting at the MIS connecting phase.
  - If the new neighborhood contains no *Dominators*, ϑ becomes a *Dominator*, broadcasts *Dominator* message, and applies the CDS algorithm locally starting at the MIS connecting phase.
  - If no *Dominator* in the new neighborhood use ϑ to reach other *Dominators*, ϑ changes to *Dominatee*, broadcasts a *Dominatee* message followed by LST_1 message.

Some remarks about Alzoubi algorithm

Even though it is the only well documented algorithm among its rivals, the designers of Alzoubi algorithm do not specify the exact criteria by which a *Dominator* node ϑ selects nodes from its *1-hop* neighbors to work as *Connectors*. The selection approach used in this thesis is based on *node degree*: the *Connector* node ν that offers paths to the maximum number of *2-hop* and *3-hop Dominators* is selected. Another remark is about the case in which two *Dominators* come into the communication range of each other. According to [4]:

> When a dominator node u joins a neighborhood, if the new set of neighbors has at least one dominator, u becomes a dominatee, updates its lists and sets, and sends a DOMINATEE message, followed by an *list1* message. Otherwise, u maintains its state as a dominator, updates its lists and sets, and sends a DOMINATOR message.

However, the fact that all nodes are mobile makes it difficult for any two nodes who recently come in contact with each other to decide which node is the one who joined the neighborhood of the other. In this thesis, this conflict is resolved as follows:

When two Dominators or more come in contact with each other, the one with the smallest ID keeps its role as a Dominator and the other one becomes a Dominatee, updates its lists and sets, and sends a Dominatee message, followed by list1 message.

4.3 General Remarks

The following remarks apply to the implementation of all algorithms studied in this research.

- Neighbors discover each other by exchanging *Hello* messages, every *Hello* massage contains the sender's ID .
- To reduce the effect of transient connections, node ϑ adds node υ to its list of neighbors after the reception of the third consecutive *Hello* messages from υ. In the same manner, node ϑ removes node υ from its neighbors list if it does not hear from it for four consecutive *Hello* message periods.
- To reduce the effect of *collisions*; after sending any message, node ϑ expects to receive a confirmation message from every intended recipient, if ϑ does not receive the confirmation within certain time, it retransmits the same message again. ϑ retransmits the unconfirmed message for up to three times.

Chapter 5

Simulation Setup and Results

This chapter describes the simulations conducted and gives the specifications of the environment in which they were conducted, explains the methodology used to process the results gathered from these simulations, presents the outcome of the processed results, and discusses these results.

5.1 Simulation setup

The simulator used in this research is the *ns-2* network simulator [44]; a widely used discrete event simulator targeted at network research. The fact that *ns-2* offers realistic models for signal propagation time, signal power attenuation, and realistic medium access protocols adds to the creditability of the simulations conducted using it. Many versions of the *ns-2* simulator are in use nowadays, the version used in this research is the *ns-2.30* installed on Fedora Core 5 running on Toshiba Satellite A100 laptop.

Two medium access control protocols, MAC protocols, were used in the simulations: the ideal MAC protocol, and the *IEEE* 802.11 MAC protocol [55]. While the ideal MAC protocol adheres to the design assumption made by the designers of the studied algorithms regarding the guaranteed delivery of broadcasted packets, the *IEEE* 802.11 MAC violates this assumption since broadcasted packets may not be correctly delivered due to packet collisions. Even though no real-world MAC protocol is ideal, the use of a *virtual* ideal MAC gives us the ability to investigate how the real-world performance of these algorithms deviate from their ideal-conditions counterparts.

To investigate the effect of the mobility model on the performance of each algorithm, two different mobility models were used with each MAC protocol: the *Random Waypoint* mobility model [61], and the *Freeway* mobility model [11]. In the *Random Waypoint* model, every node, ϑ, has an initial position $P_i = (x_i, y_i)$ at which it stays for a given period of time called the *Pause Time*; at the end of this period, ϑ moves toward a randomly chosen destination $P_d = (x_d, y_d)$ with a constant speed V. ϑ stays at P_d for another pause time at the end of which it moves to another destination and so on. Both of the speed and the pause time are uniformly distributed in the intervals $[V_{min}, V_{max}]$ and $[T_{min}, T_{max}]$ respectively. The simulations based on this model were run over a square field of 2000 m $\times$ 2000 m. The number of nodes used ranges from 50 to 150 with increments of 10. For every number of nodes, three different ranges of speed were employed: 15-20, 20-25, 25-30 meter per second. For every speed and number of nodes, six different movement scenarios are generated, and every scenario is used in one simulation, the average of the results of these six simulations is calculated and used to represent the algorithm's behavior under the given speed and number of nodes.

The Freeway mobility model emulates the mobility behavior of vehicles on freeways. In this model, the direction of node movement is controlled by the direction of the lane on which it moves; however, the speed of node movement is controlled by the following relations:

$$|S_i\left(t+1\right)| = |S_i\left(t\right)| + random\left(\right) * |a_i\left(t\right)| \tag{5.1}$$

if node jf node j is in front of node i

$$\forall t\ D_{i,j}\left(t\right) \leq SD \Rightarrow |S_i\left(t\right)| \leq |S_j\left(t\right)| \tag{5.2}$$

Where $S_i\left(t\right)$ and $a_i\left(t\right)$ are the speed and the acceleration of node i at time t, $D_{i,j}\left(t\right)$ is the distance between nodes i and j at time t, and SD is the safety distance. The simulations based on this model were run over an field of 3000 m by 1000 m divided into ten parallel lanes. The number of nodes used in these simulations ranges from 50 to 175 with increments of 25. For every number of nodes, three different ranges of speed were employed : 15-20, 20-25, 25-30 m/s. For every speed and number of nodes, three different movement scenarios are generated, and every scenario is used in one simulation, the average of the results of the three simulations is calculated and used to represent the algorithm's behavior under the given speed and number of nodes. In all the simulations conducted in this research, every simulation runs for 300 seconds, the radio transmission range of every node is set to 250 meters and the *Hello period* is set to 1 second.

5.2 Simulation results

This section presents the results of the simulations conducted in this research. These results are presented according to the mobility model employed.

5.2.1 Simulations based on the Random Waypoint mobility model

The results of the simulations based on this mobility model are divided into two groups. The first group includes the results obtained using a virtual ideal MAC protocol and the second presents the results obtained using a real MAC protocol: the *IEEE802.11* MAC.

- Results obtained using ideal MAC

The following figures compare the performance of the studied algorithms in terms of CDS size, CDS establishment time [1], the total running time [2], the average number of bytes transmitted by every node, the average number of bytes broadcasted, and the average number of bytes unicasted by every node. Moreover, they show the effect of the speed of movement on the performance of each algorithm.

- **The CDS size**

The average size of the CDS produced by each algorithm as a function of the number of nodes are given in figures 5.1, 5.2, and 5.3 for the three ranges of speed. The most important remark about these figures is that Alzoubi algorithm consistently produces the smallest CDS; moreover, Alzoubi algorithm is the best in terms of scalability and the MPR is the worst. In fact, the number of nodes that can be accommodated by the MPR algorithm depends on the speed of movement, as the speed increases, the number of nodes decreases. Another remark is that the size of the CDS produced of the studied algorithms increases linearly with the number of nodes; however, the rates of increase are different. Finally, it can be noticed that the increase in the speed of movement is associated with CDS size decrease. This decrease in the CDS size results from the fact that fast moving nodes encounter more frequent topology changes, these changes force many CDS members to go back and forth in and out of the CDS; therefore, the average number of stable CDS members, i.e. CDS size, gets smaller.

[1]This is the time at which every node knows whether it is a CDS member or not.

[2]Time spent during CDS creation and maintenance phases.

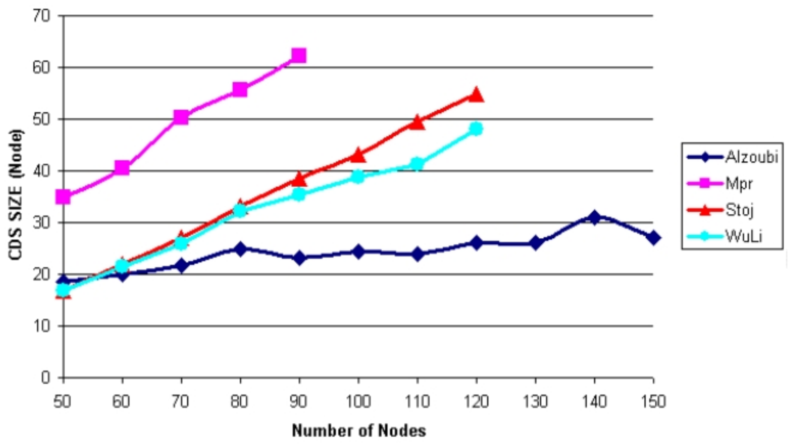

Figure 5.1: CDS size at speed range 15-20 m/s.

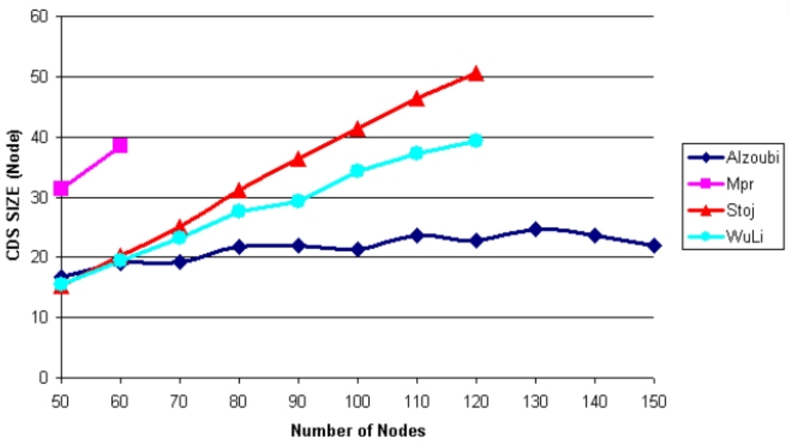

Figure 5.2: CDS size at speed range 20-25 m/s.

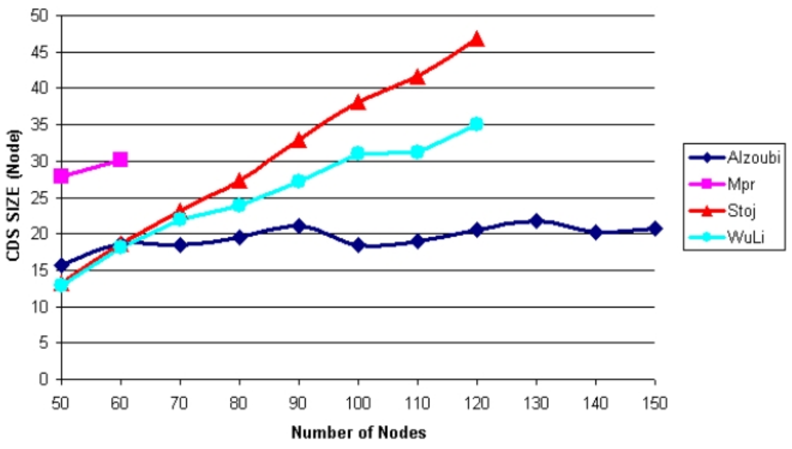

Figure 5.3: CDS size at speed range 25-30 m/s.

- **The CDS establishment time**

Figures 5.4, 5.5, and 5.6 show the average CDS establishment time for each algorithm as a function of the number of nodes, for three ranges of speed. It can be noticed from these figures that Stojmenovic algorithm is the fastest algorithm in constructing the CDS and the MPR algorithm is the slowest. The fact that there is an internodal dependency in all of the studied algorithms except for Stojmenovic algorithm is the reason behind their long CDS establishment times.

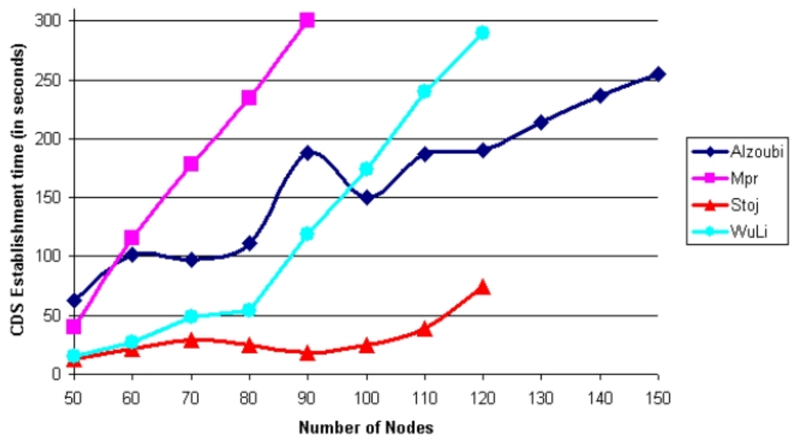

Figure 5.4: CDS establishment time at speed range 15-20 m/s.

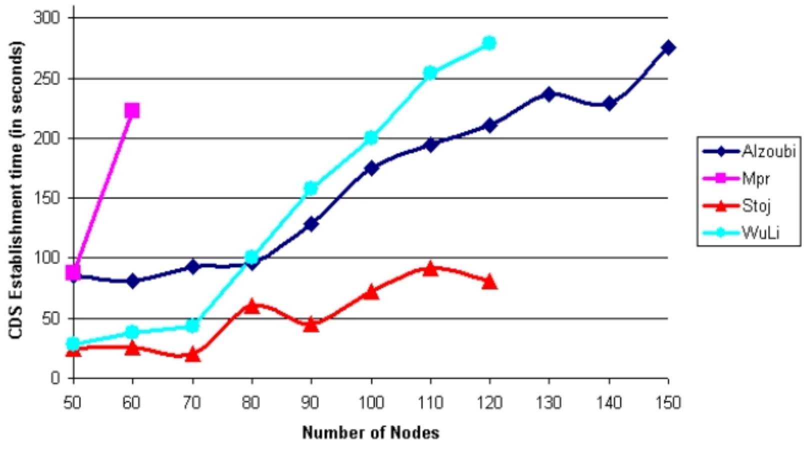

Figure 5.5: CDS establishment time at speed range 20-25 m/s.

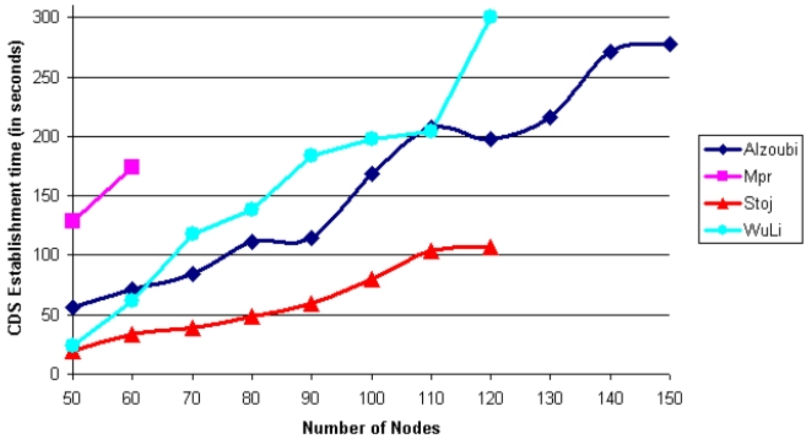

Figure 5.6: CDS establishment time at speed range 25-30 m/s.

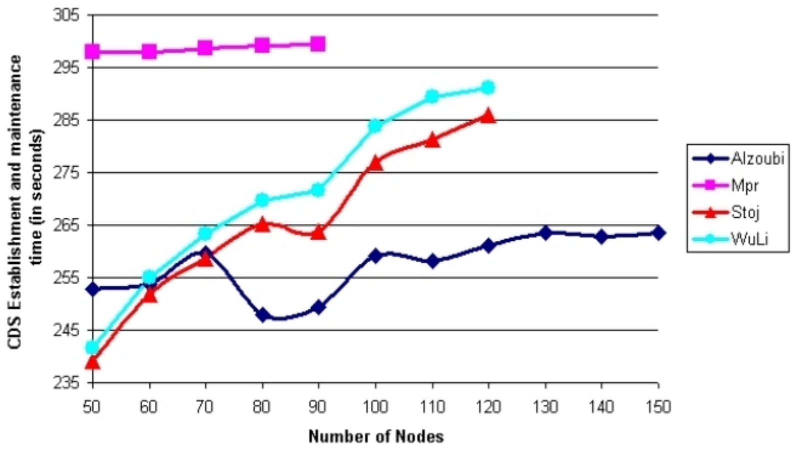

Figure 5.7: The total running time at speed range 15-20 m/s.

- **The total running time**

The average total time of each algorithm as a function of the number of nodes for each of the three speed ranges are given in figures 5.7, 5.8, and 5.9. Based on these figures, it can be concluded that Alzoubi algorithm produces the most stable CDS among the studied algorithms; this conclusion is based on the fact that even though Alzoubi algorithm is not the fastest in the CDS construction phase, its total running time is the shortest compared to its rivals. It can also be concluded that although Stojmenovic variant of Wu and Li algorithm improves on the original algorithm, this improvement is minimal (less than 4% of the total running time).

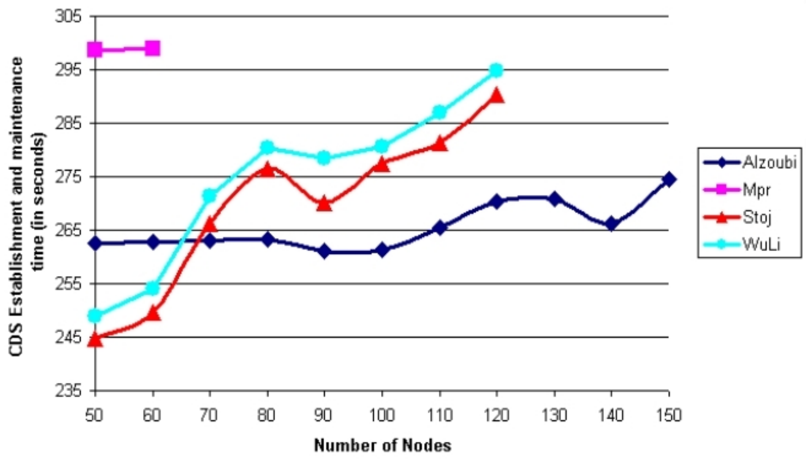

Figure 5.8: The total running time at speed range 20-25 m/s.

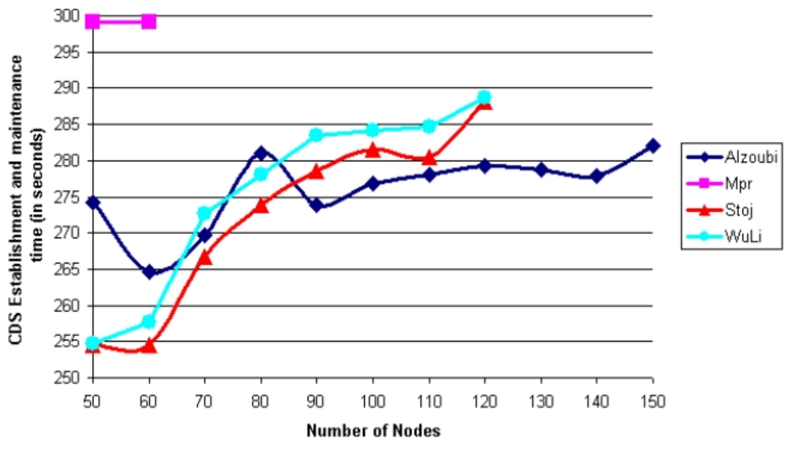

Figure 5.9: The total running time at speed range 25-30 m/s.

- **The total number of transmitted bytes**

Figures 5.10, 5.11, and 5.12 show the average total number of bytes transmitted either broadcasting or by unicasting during the CDS creation and maintenance phases of each algorithm as a function of the number of nodes and for three ranges of speed. The important remark to be made here is about the fast growth of the total number of bytes transmitted by Alzoubi algorithm with the increase in the number of nodes. The other remark that can be made is that the average total numbers of bytes transmitted by Stojmenovic, Wu and Li, and the MPR algorithms are almost the same. Finally, while the total number of transmitted bytes is not sensitive to the mobility rate, i.e., speed range, for Stojmenovic, Wu and Li, and the MPR algorithms, it decreases as the speed increase for Alzoubi algorithm.

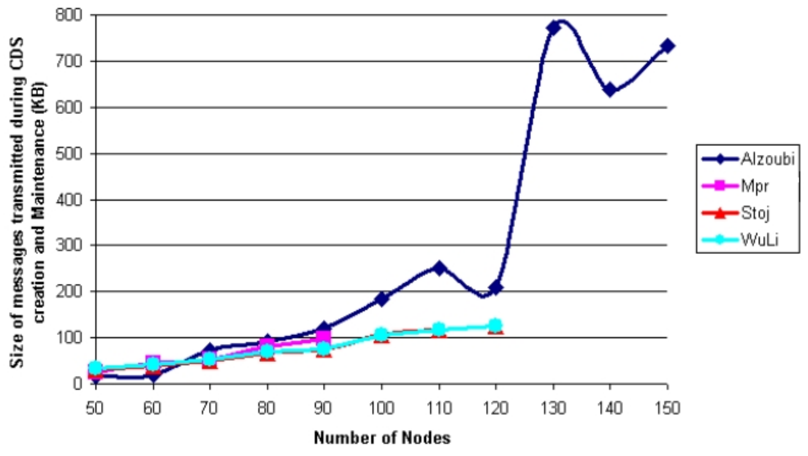

Figure 5.10: The total number of bytes transmitted at speed range 15-20 m/s.

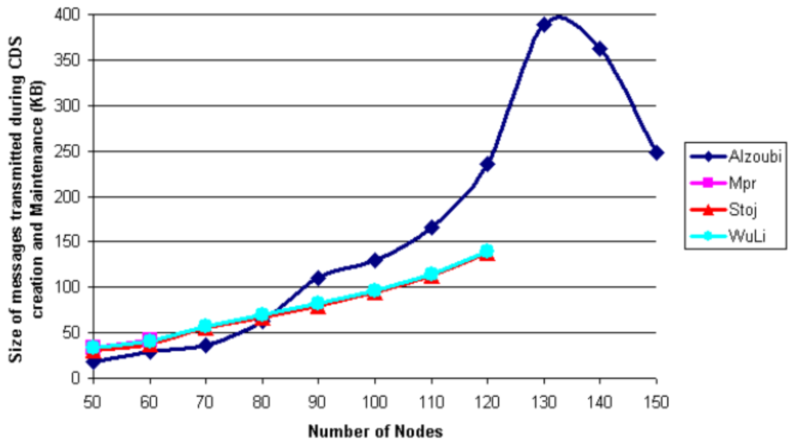

Figure 5.11: The total number of bytes transmitted at speed range 20-25 m/s.

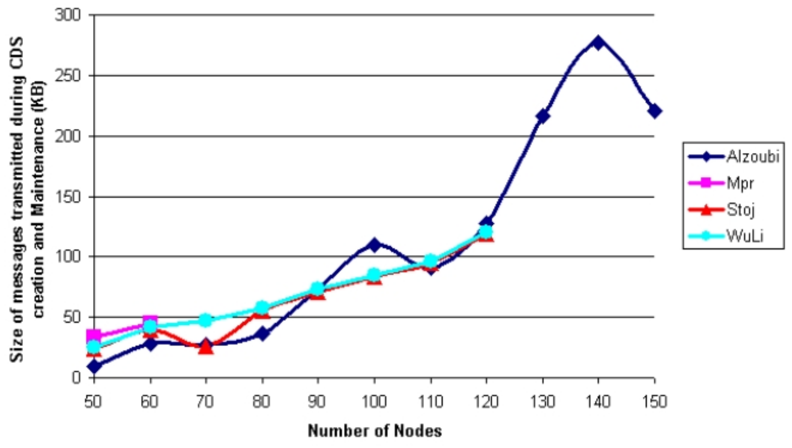

Figure 5.12: The total number of bytes transmitted at speed range 25-30 m/s.

- **The total number of broadcasted bytes**

In MANETs, messages sent by broadcasting face higher risk of collision than those sent by unicast. The tendency of any algorithm to use message broadcast is directly related to its applicability in MANETs, the more the broadcasting tendency the less is the applicability potential. Figures 5.13, 5.14, and 5.15 show the average total number of bytes broadcasted during the CDS creation and maintenance phases of each algorithm as a function of the number of nodes for three ranges of speed. Based on these figures, one can conclude that Alzoubi algorithm is the most suitable among all the studied algorithms for MANETs applications, since the total number of bytes broadcasted in this algorithm is much smaller than those broadcasted by other algorithms.

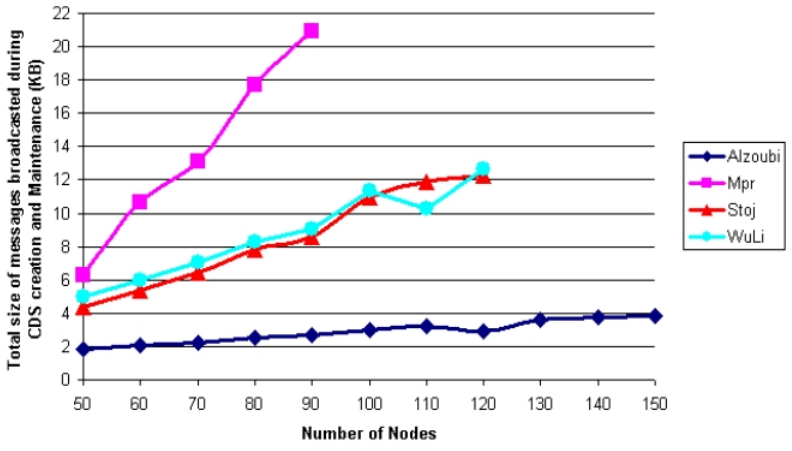

Figure 5.13: The total number of bytes broadcasted at speed range 15-20 m/s.

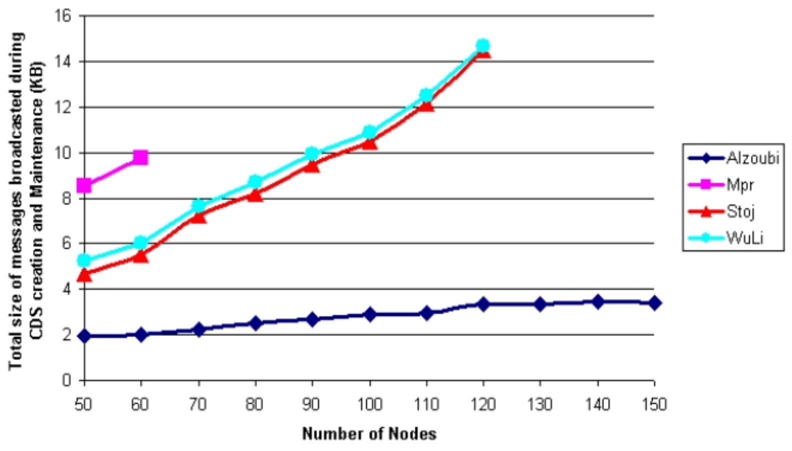

Figure 5.14: The total number of bytes broadcasted at speed range 20-25 m/s.

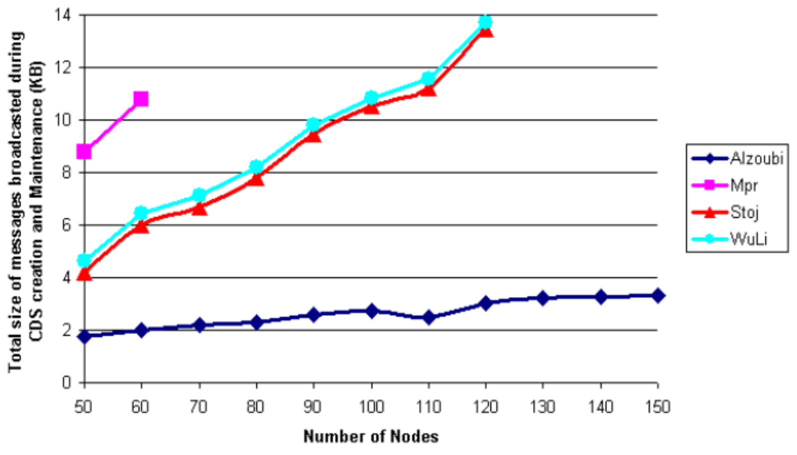

Figure 5.15: The total number of bytes broadcasted at speed range 25-30 m/s.

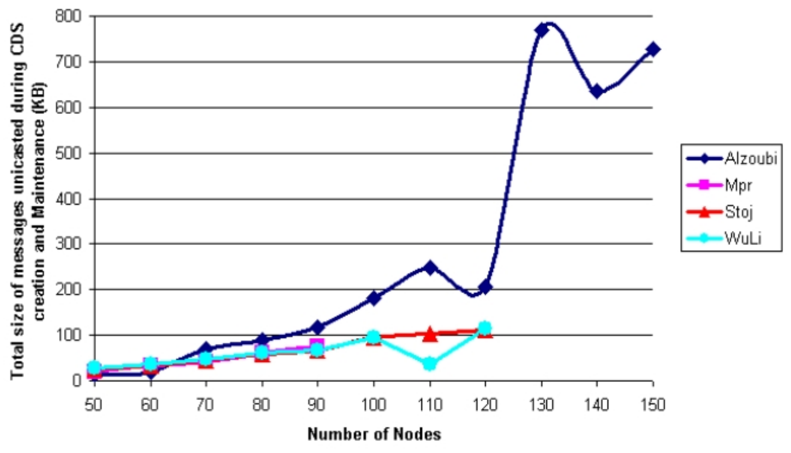

Figure 5.16: The total number of bytes unicasted at speed range 15-20 m/s.

- **The total number of unicasted bytes**

Figures 5.16, 5.17, and 5.18 show the average total number of bytes unicasted during the CDS creation and maintenance phases of each algorithm as a function of the number of nodes, for three ranges of speed. It is clear that the average total number of bytes unicasted by Alzoubi algorithm is the largest among all the studied algorithms.

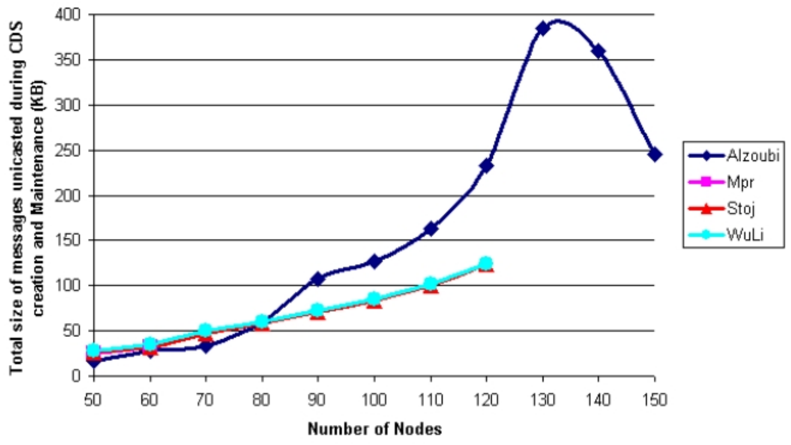

Figure 5.17: The total number of bytes unicasted at speed range 20-25 m/s.

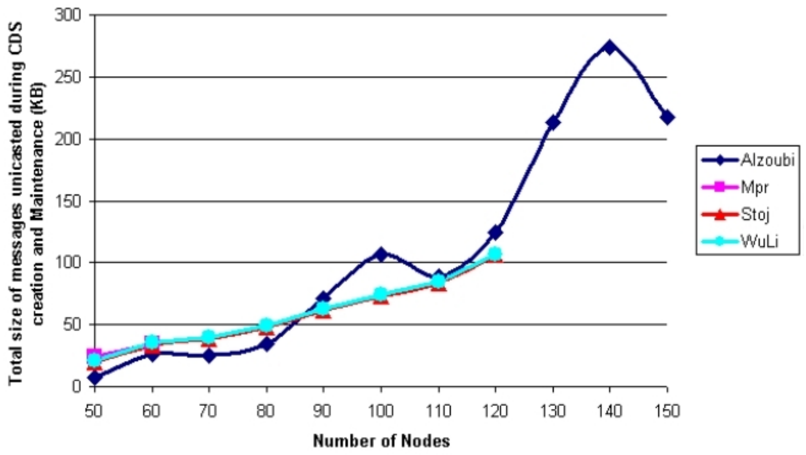

Figure 5.18: The total number of bytes unicasted at speed range 25-30 m/s.

- Results obtained using the *IEEE 802.11 MAC*

Due to the fact that the IEEE 802.11 MAC protocol [55] does not offer any mechanism that guarantees the delivery of messages sent by broadcast, all of the CDS creation algorithms that rely on broadcasting to establish and maintain the CDS, namely: Wu and Li algorithm, Stojmenovic algorithm, and the MPR algorithm, are not able to function properly using this MAC, i.e., they are not applicable in the real world. On the other hand, the moderate use of broadcasting and the use of unicasting are the reasons behind the ability of Alzoubi algorithm to function using the IEEE 802.11 MAC protocol with some qualifications. In addition to the loss of messages due collisions, the fact that the IEEE 802.11 MAC protocol forces all nodes that are located in the transmission range of any currently transmitting or receiving node to delay transmission and to buffer any packet that is ready for transmission, and because of the limited-capacity of the transmission buffers, messages are lost due to the overflow of these buffers. As a result of these message losses, Alzoubi algorithm fails in keeping track of topology changes and in updating nodes' roles in response to these topology changes. Therefore, the performance of Alzoubi algorithm based on the use of the IEEE 802.11 MAC protocol deviates significantly from its performance based on the use of the virtual ideal MAC protocol. The following figures show the deviation in the performance of Alzoubi algorithm based on the use of the IEEE 802.11 MAC protocol from that based on the use of the ideal MAC protocol for three ranges of speed.

- **Deviation in CDS size**

The average size of the CDS produced by Alzoubi algorithm based on the use of the IEEE 802.11 MAC protocol and the corresponding average CDS size produced by the virtual ideal MAC protocol as a function of the number of nodes, and the effect of the speed of movement on these CDS sizes are shown in figures 5.19, 5.20, and 5.21. The important remark that can be made about these figures is that the deviation in the CDS size increases with the increase in the number of nodes; the reason is that the increase in the number of nodes leads to an increase in the rate of packet loss; consequently, the inability of nodes to keep track of topology changes and to update their roles accordingly becomes more apparent.

- **Deviation in CDS establishment time**

Figures 5.22, 5.23, and 5.24 show how the CDS establishment time of Alzoubi algorithm based on the IEEE 802 MAC deviates from that based on the virtual ideal MAC. Here too, the deviation increases with the increase in the MANET size.

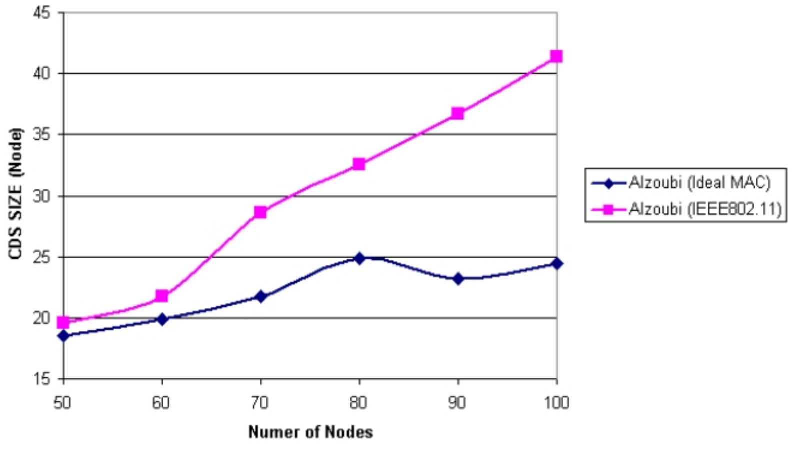

Figure 5.19: Deviation in CDS size at speed range 15-20 m/s.

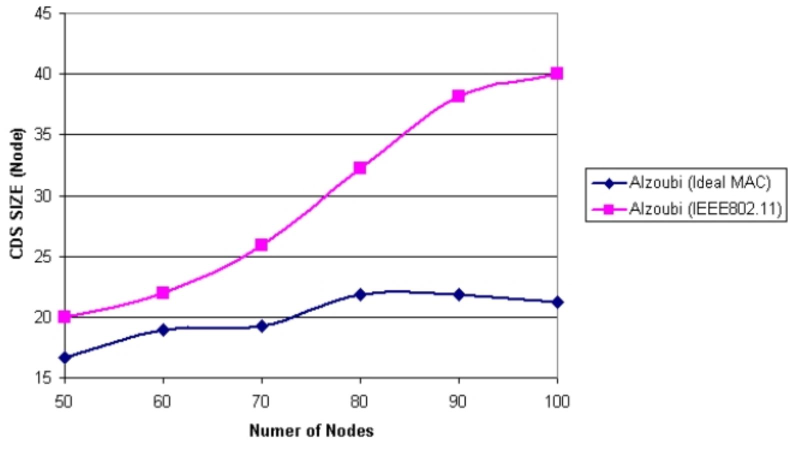

Figure 5.20: Deviation in CDS size at speed range 20-25 m/s.

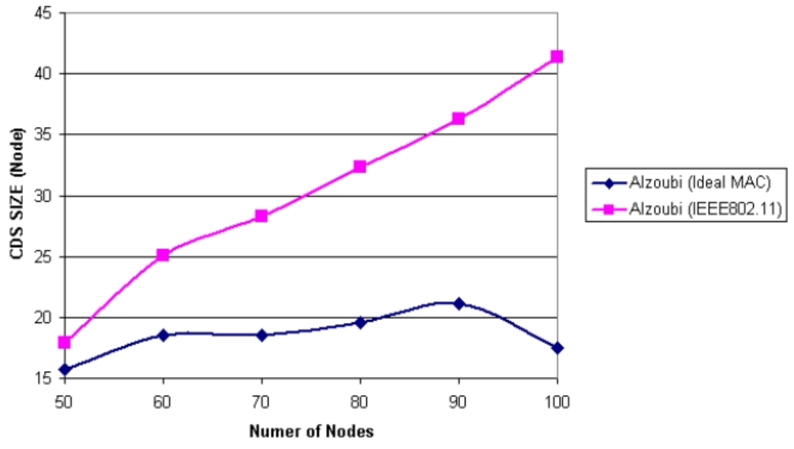

Figure 5.21: Deviation in CDS size at speed range 25-30 m/s.

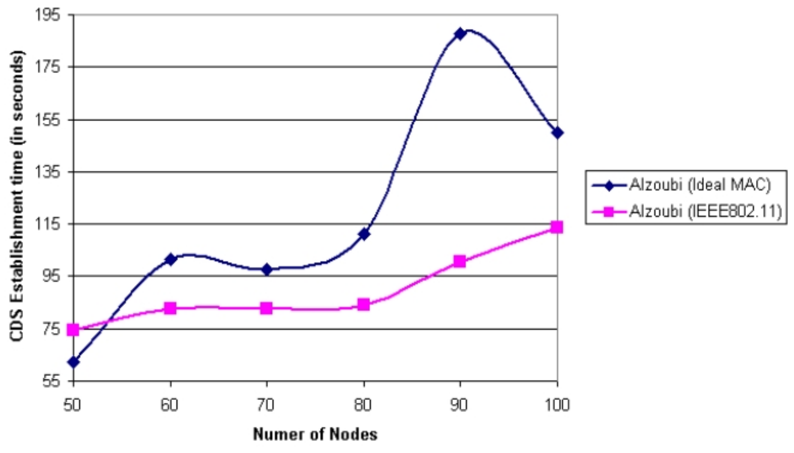

Figure 5.22: Deviation in CDS establishment time at speed range 15-20 m/s.

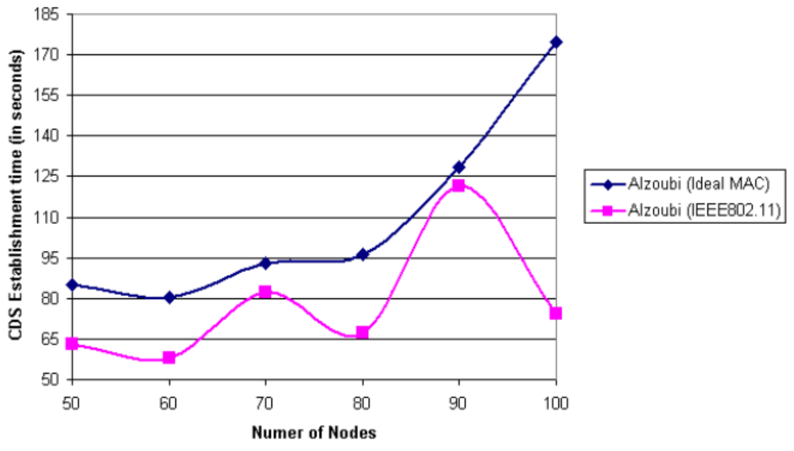

Figure 5.23: Deviation in CDS establishment time at speed range 20-25 m/s.

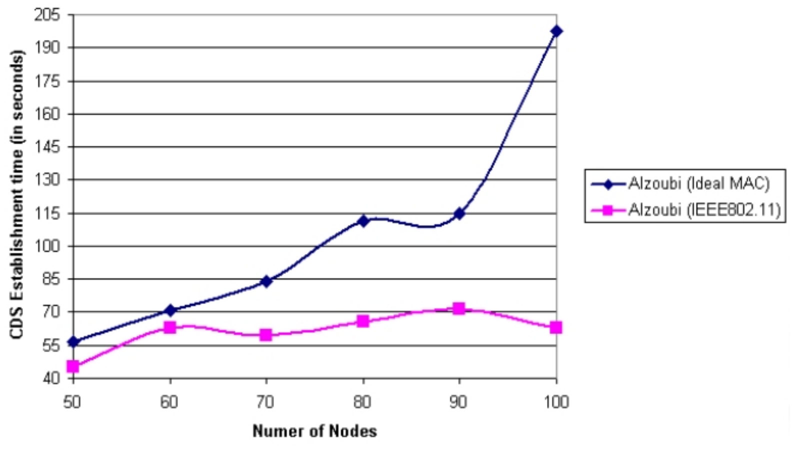

Figure 5.24: Deviation in CDS establishment time at speed range 25-30 m/s.

- **Deviation in the total running time**

The deviation in the total running time as a function of the number of nodes for each of the three ranges of speed is given in figures 5.25, 5.26, and 5.27. It is obvious that the total running time based on the use of the IEEE 802.11 MAC protocol is longer than that based on the use of the virtual ideal MAC. Since the CDS establishment time based on the use of the IEEE 802.11 MAC is shorter than that based on the use of the virtual ideal MAC, one can conclude that the CDS maintenance time based on the use of the IEEE 802.11 MAC is longer than that based on the use of the virtual ideal MAC. The need of neighboring nodes to exchange messages with each other in order to maintain the CDS, and the fact that there might be a need to transmit the same message many times to compensate for message losses due to collisions, are the reasons behind the longer time spent in maintaining the CDS based on the use of the IEEE 802.11 MAC protocol.

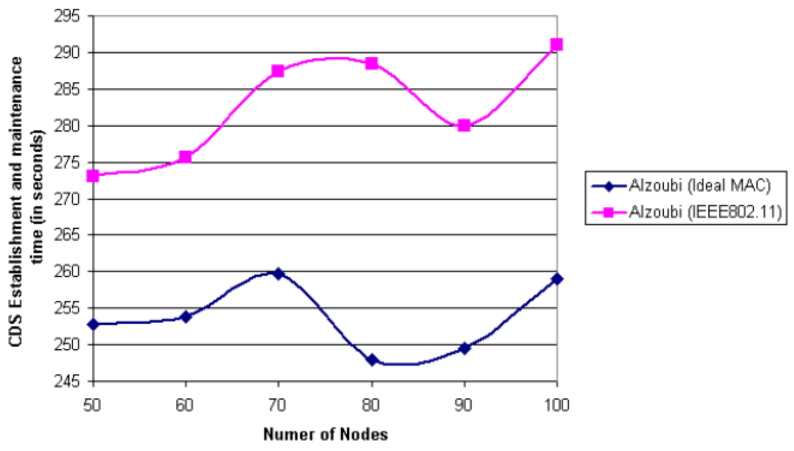

Figure 5.25: The deviation in the total running time at speed range 15-20 m/s.

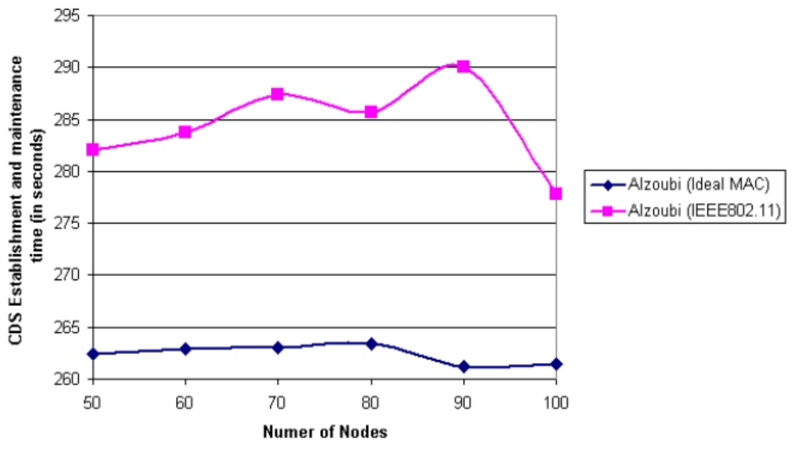

Figure 5.26: The deviation in the total running time at speed range 20-25 m/s.

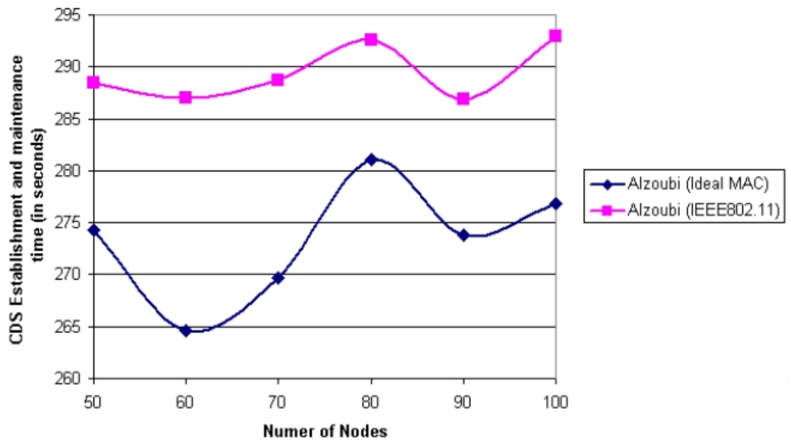

Figure 5.27: The deviation in the total running time at speed range 25-30 m/s.

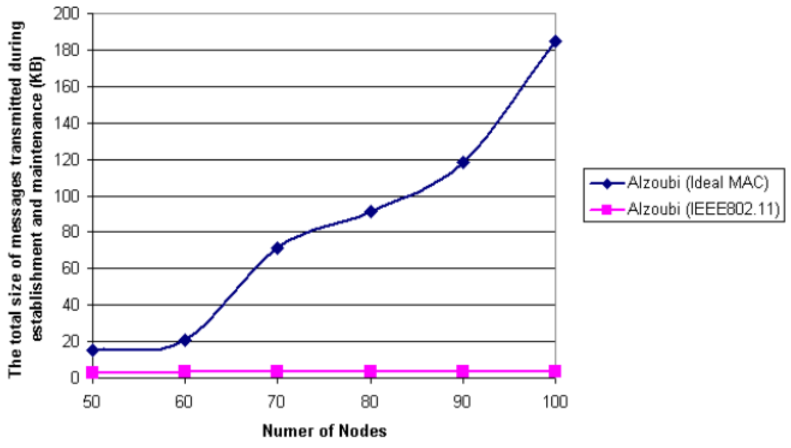

Figure 5.28: The deviation in the number of transmitted bytes at 15-20 m/s.

- **Deviation in the total number of transmitted bytes**

Figures 5.28, 5.29, and 5.30 show the deviation in the total number of transmitted bytes as a function of the number of nodes for each of the three ranges of speed. This large deviation represents the difference between the bandwidth required by Alzoubi algorithm and the bandwidth that can be granted by the wireless communication channel. This is the underlying reason for the deterioration of Alzoubi algorithm performance under real-world conditions.

- **Deviation in the total number of broadcasted bytes**

The deviation in the total number of broadcasted bytes as a function of the number of nodes for each of the three ranges of speed is shown in figures 5.31, 5.32, and 5.33. It can be noticed that the wireless communication channel saturates at

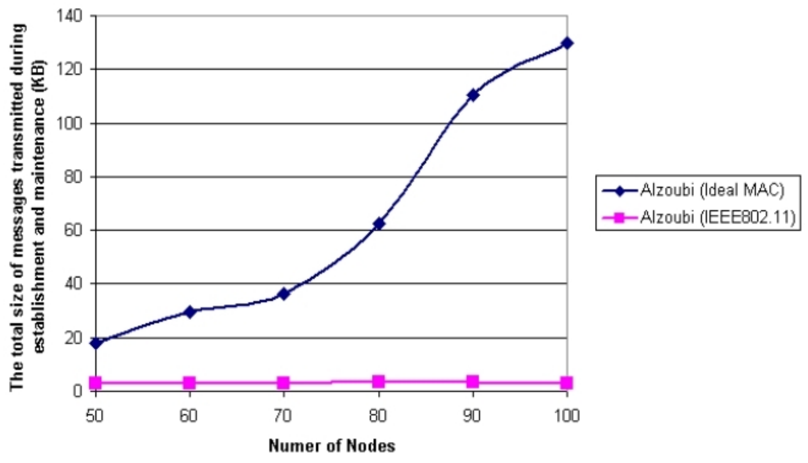

Figure 5.29: The deviation in the number of transmitted bvtes at 20-25 m/s.

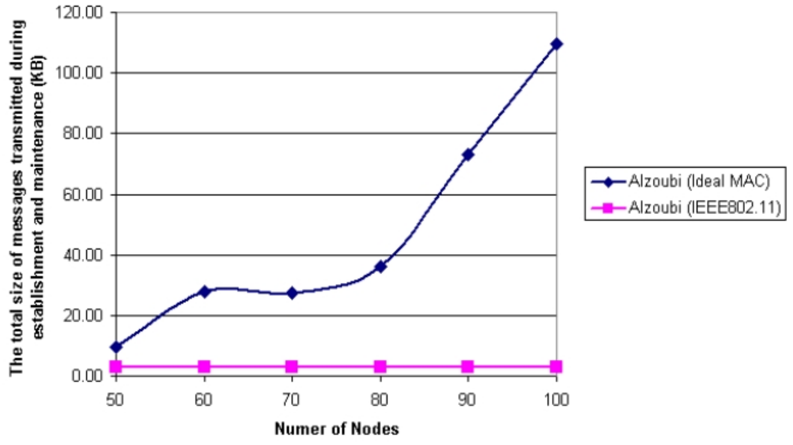

Figure 5.30: The deviation in the number of transmitted bytes at 25-30 m/s.

1.4 KB. As a result, it fails to satisfy the broadcast demands of Alzoubi algorithm, which exceeds 1.7 KB for all of the considered cases.

- **Deviation in the total number of unicasted bytes**

Figures 5.34, 5.35, and 5.36 show the deviation in the total number of unicasted bytes as a function of the number of nodes for each of the three ranges of speed. Due to the fact that Alzoubi algorithm uses unicasting more than broadcasting, it can be noticed that the deviation in the total number of unicasted bytes is larger than the deviation in total number of broadcasted bytes.

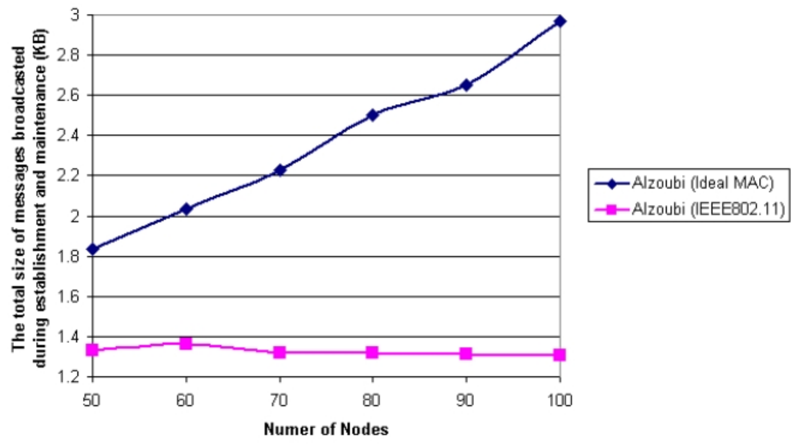

Figure 5.31: The deviation in the number of bytes broadcasted at 15-20 m/s.

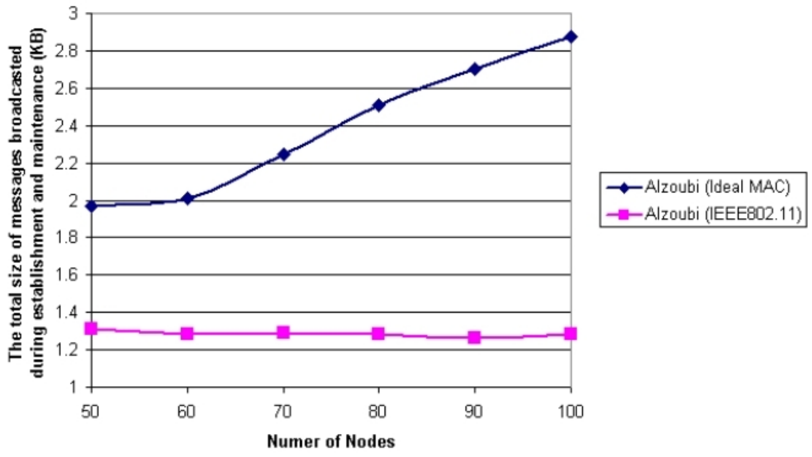

Figure 5.32: The deviation in the number of bytes broadcasted at 20-25 m/s.

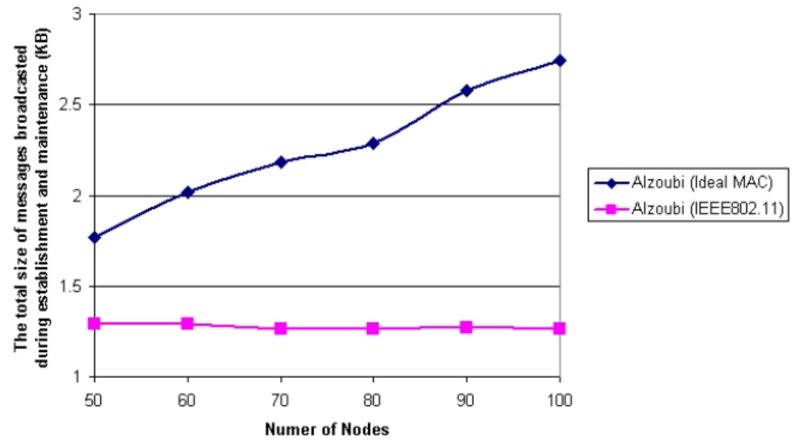

Figure 5.33: The deviation in the number of bytes broadcasted at 25-30 m/s.

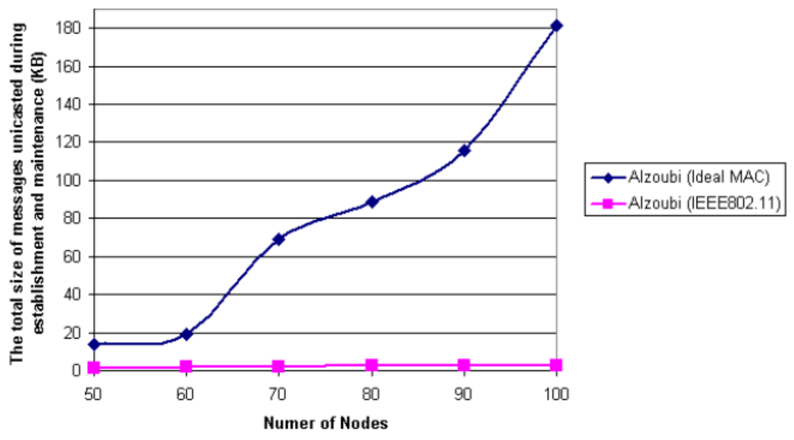

Figure 5.34: The deviation in the number of bytes unicasted at 15-20 m/s.

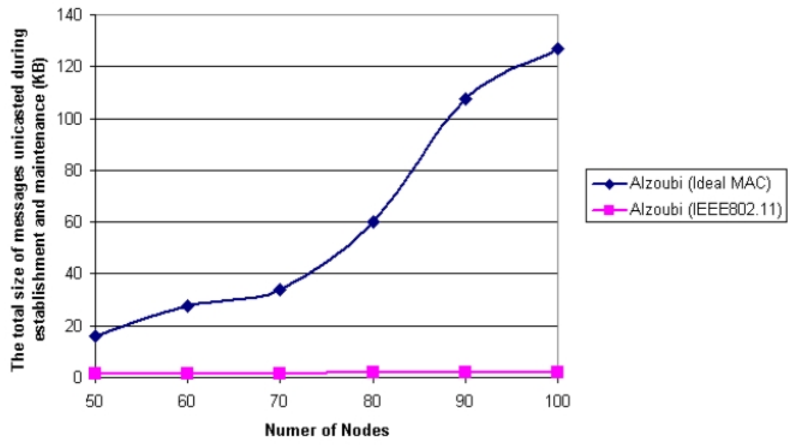

Figure 5.35: The deviation in the number of bytes unicasted at 20-25 m/s.

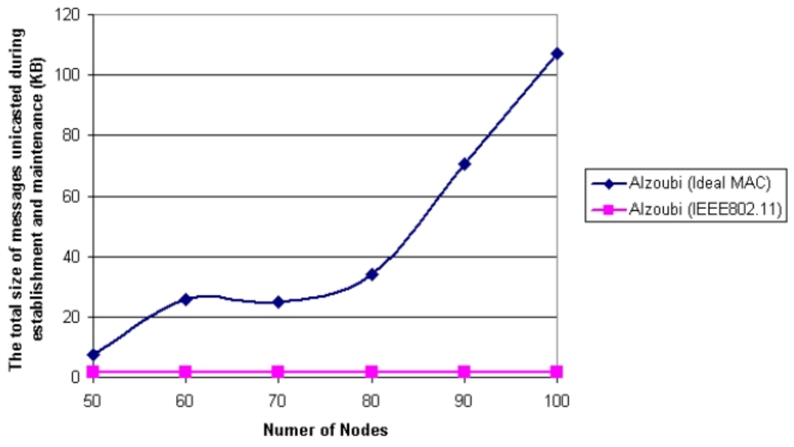

Figure 5.36: The deviation in the number of bytes unicasted at 25-30 m/s.

5.2.2 Simulations based on the Freeway mobility model

The results of the simulations based on this mobility model are divided into two groups. The first group includes the results obtained using a virtual ideal MAC protocol and the second presents the results obtained using the *IEEE802.11* MAC.

- Results obtained using ideal MAC

The following figures compare the performance of the studied algorithms in terms of CDS size, CDS establishment time, the total running time, the average total number of bytes transmitted, the average number of bytes broadcasted, and the average number of bytes unicasted by every node. Moreover, they show the effect of the speed of movement on the performance of each algorithm.

- **The CDS size**

The average size of the CDS produced by each algorithm as a function of the number of nodes for the three ranges of speed are shown in figures 5.37, 5.38, and 5.39. The same remarks as those made about the CDS size of the Random Waypoint mobility model can be made here. It is obvious that most of the time Alzoubi algorithm produces the smallest CDS and it is the best in terms of scalability. It can be noticed that the number of nodes accommodated by the MPR algorithm depends on the speed of movement, as the speed increases the number of nodes decreases. Moreover, it can be noticed that the increase in the speed of movement is associated with CDS size decrease for the other algorithms. This CDS size decrease results from the fact that fast moving nodes encounter more frequent topology changes, these changes force many CDS members to go back and forth in and out of the CDS; therefore, the average number of stable CDS members, i.e. CDS size, gets smaller.

- **The CDS establishment time**

Figures 5.40, 5.41, and 5.42 show the average CDS establishment time for each algorithm as a function of the number of nodes for three ranges of speed. Here too, as in the Random Waypoint mobility model, Stojmenovic algorithm is the fastest algorithm in constructing the CDS and the MPR algorithm is the slowest. As mentioned earlier, the fact that there is an internodal dependency in all of the studied algorithms except for Stojmenovic algorithm is the reason behind their longer CDS establishment times.

- **The total running time**

The average total time of each algorithm as a function of the number of nodes

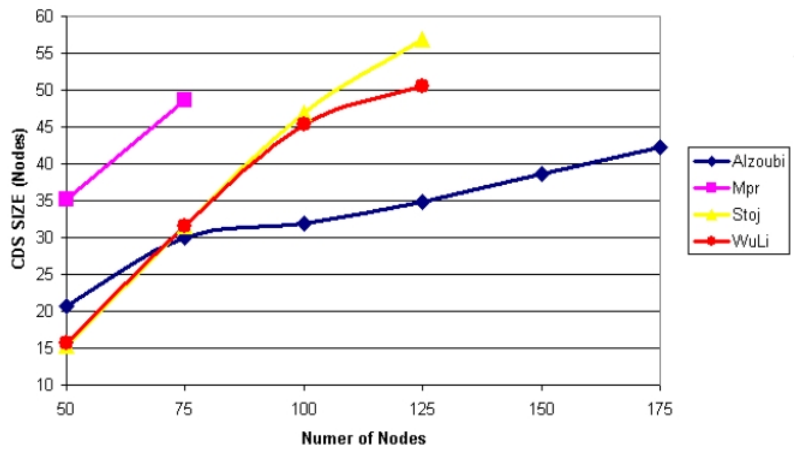

Figure 5.37: CDS size at speed range 15-20 m/s.

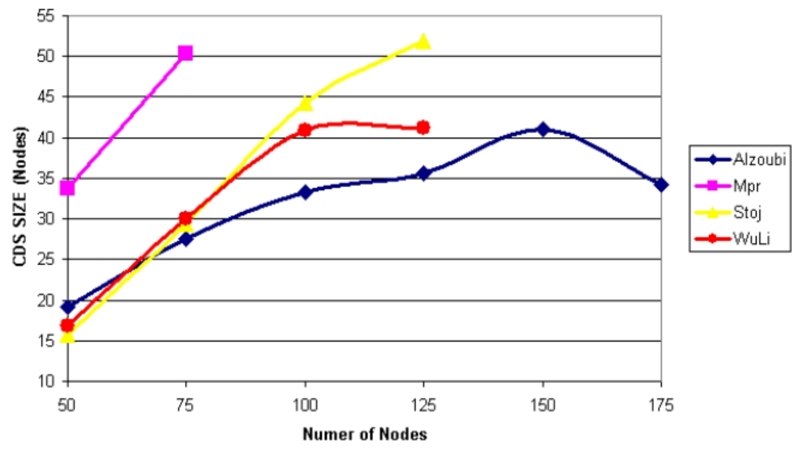

Figure 5.38: CDS size at speed range 20-25 m/s.

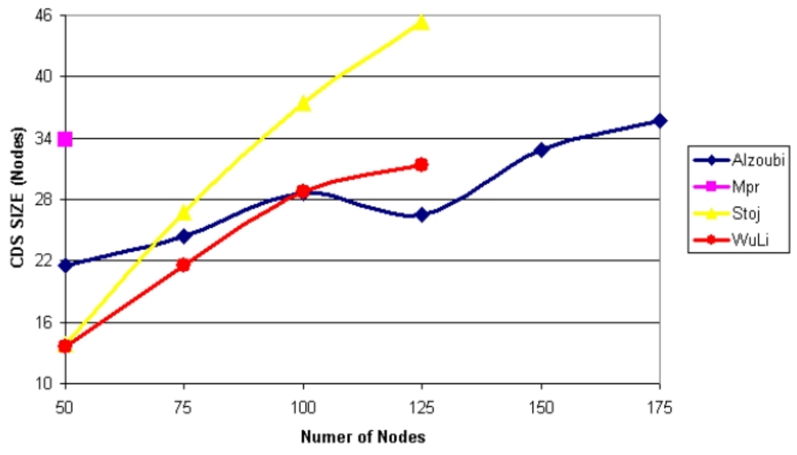

Figure 5.39: CDS size at speed range 25-30 m/s.

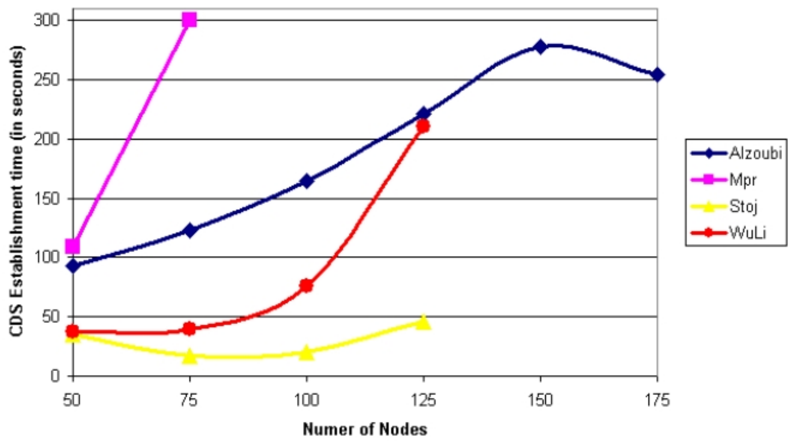

Figure 5.40: CDS establishment time at speed range 15-20 m/s.

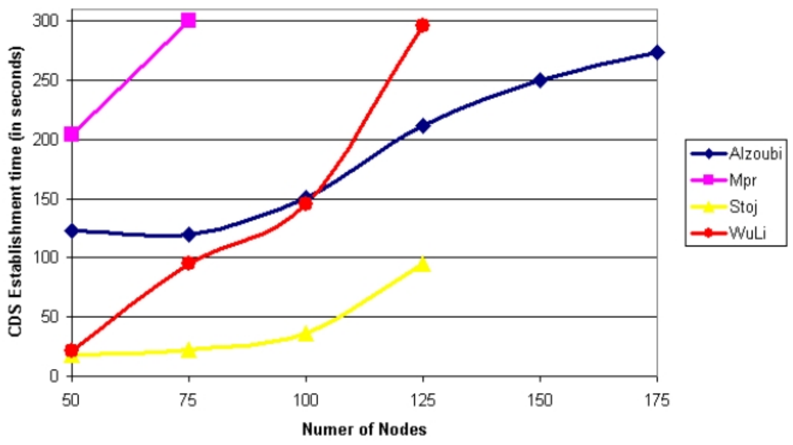

Figure 5.41: CDS establishment time at speed range 20-25 m/s.

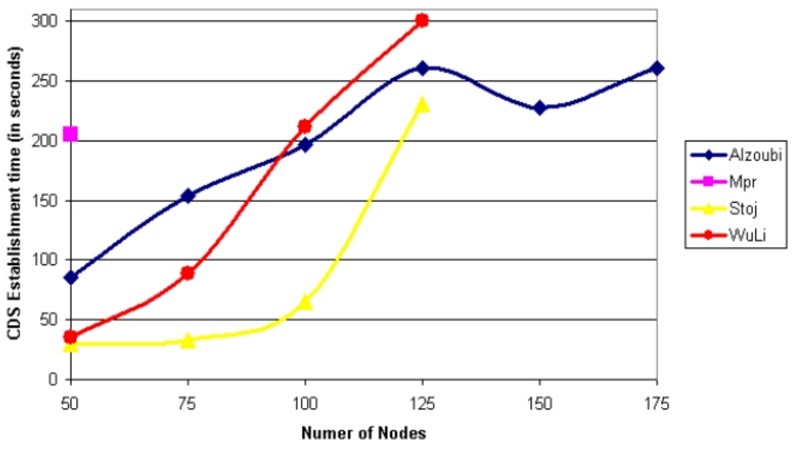

Figure 5.42: CDS establishment time at speed range 25-30 m/s.

for each of the three ranges of speed are given in figures 5.43, 5.44, and 5.45. The worthwhile remark to be made here is that the ranking of the studied algorithms according to their total running time is exactly the same as their ranking based on the total running time under the Random Waypoint mobility model. In both cases, Alzoubi algorithm has the shortest total running time, Stojmenovic algorithm has the second rank, Wu and Li algorithm comes third, and the MPR algorithm has the longest total running time. Therefore, one can conclude that Alzoubi algorithm produces the most stable CDS; this conclusion is based on the fact that even though Alzoubi algorithm is not the fastest in the CDS construction phase, its total running time is the shortest compared to its rivals.

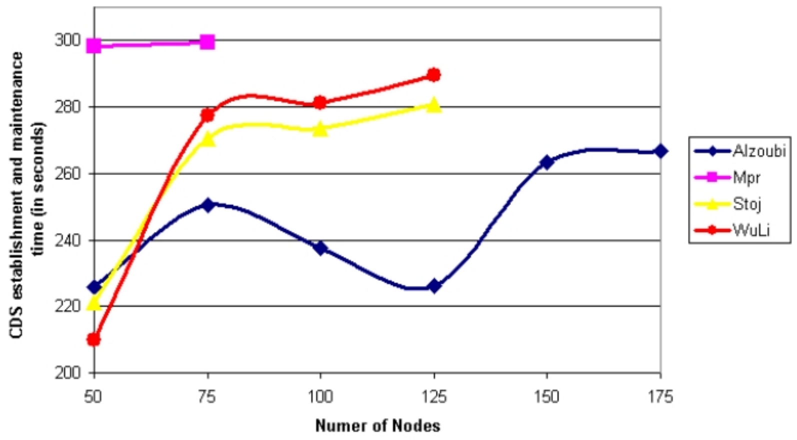

Figure 5.43: The total running time at speed range 15-20 m/s.

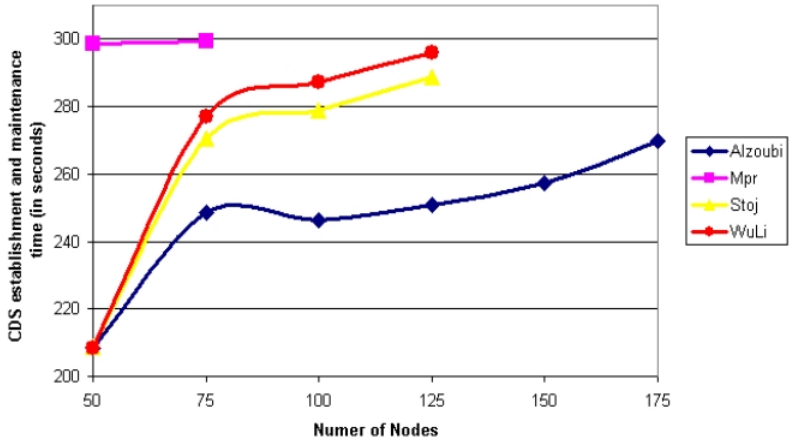

Figure 5.44: The total running time at speed range 20-25 m/s.

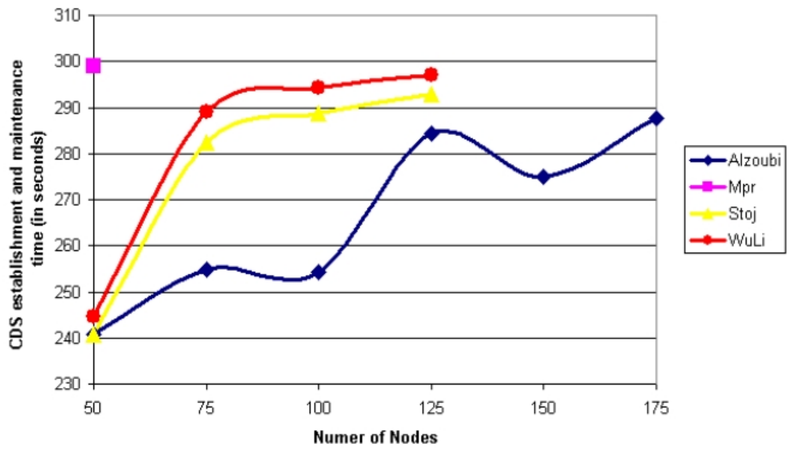

Figure 5.45: The total running time at speed range 25-30 m/s.

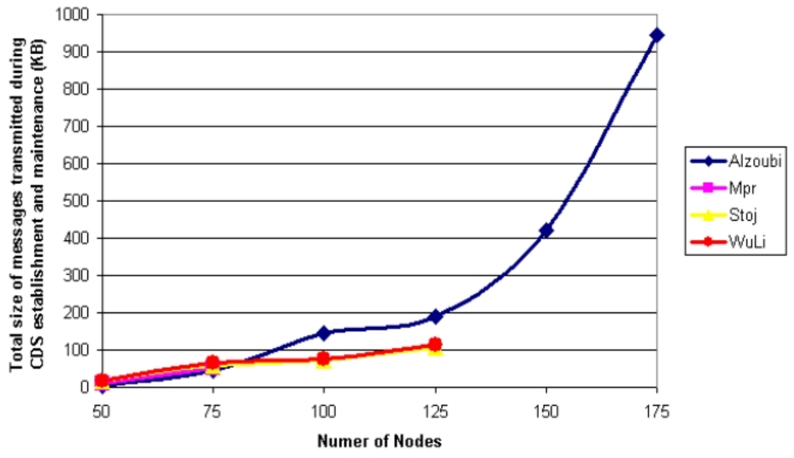

Figure 5.46: The total number of bytes transmitted at speed range 15-20 m/s.

- **The total number of transmitted bytes**

Figures 5.46, 5.47, and 5.48 show the average total number of bytes transmitted either broadcasting or unicasting during the CDS creation and maintenance phases of each algorithm as a function of the number of nodes for three ranges of speed. It is important to note the fast growth of the total number of bytes transmitted by Alzoubi algorithm with the increase in the number of nodes. The other remark that can be made is that the average total number of bytes transmitted by Stojmenovic algorithm and Wu and Li algorithm is almost the same. Note that these are the same remarks made about the total number of transmitted bytes under the Random Waypoint mobility model.

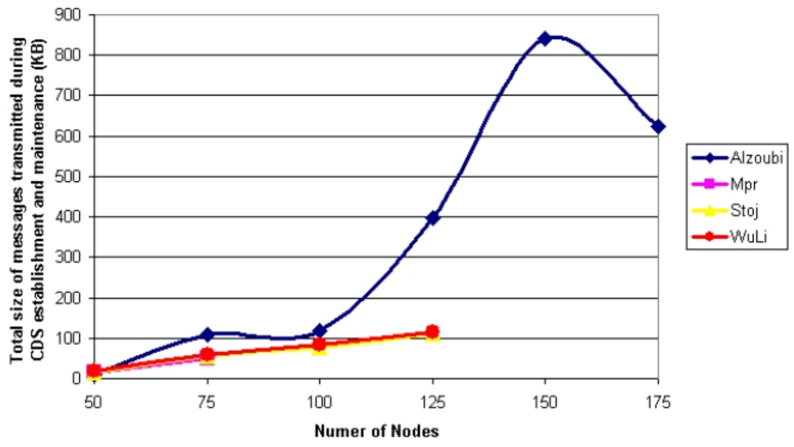

Figure 5.47: The total number of bytes transmitted at speed range 20-25 m/s.

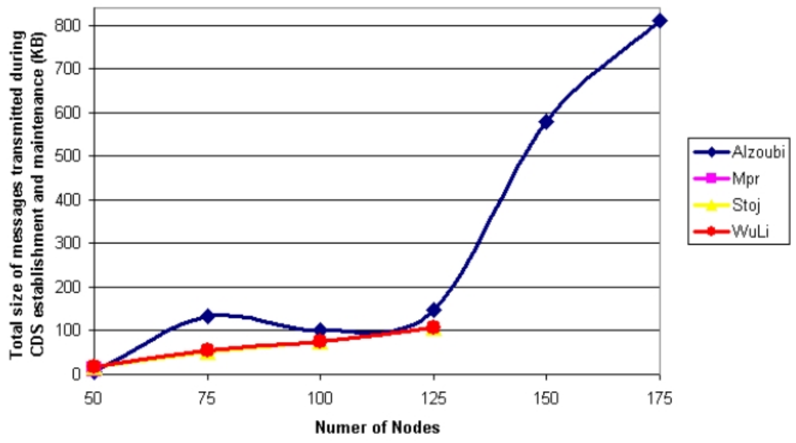

Figure 5.48: The total number of bytes transmitted at speed range 25-30 m/s.

- **The total number of broadcasted bytes**

Figures 5.49, 5.50, and 5.51 show the average total number of bytes broadcasted during the CDS creation and maintenance phases of each algorithm as a function of the number of nodes for three ranges of speed. It can be noticed here that, as in the simulations based on the Random Waypoint mobility model, the average total number of bytes broadcasted by Alzoubi algorithm is much smaller than those broadcasted by any of the other studied algorithms.

- **The total number of unicasted bytes**

Figures 5.52, 5.53, and 5.54 show the average total number of bytes unicasted during the CDS creation and maintenance phases of each algorithm as a function of the number of nodes for three ranges of speed. It can be noticed that the average

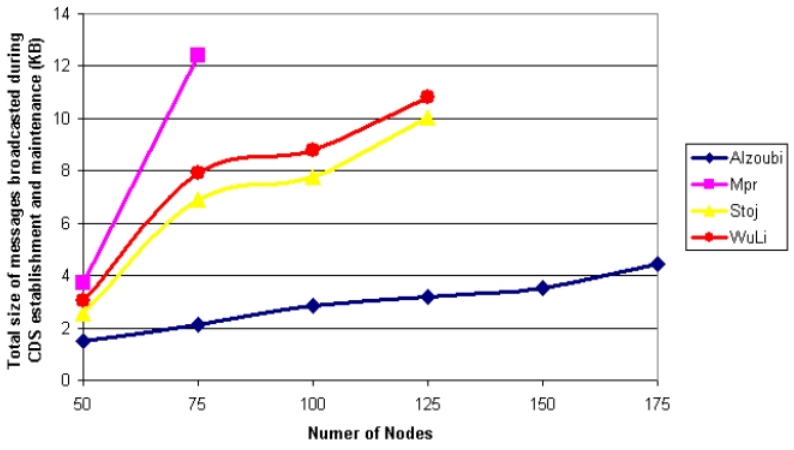

Figure 5.49: The total number of bytes broadcasted at speed range 15-20 m/s.

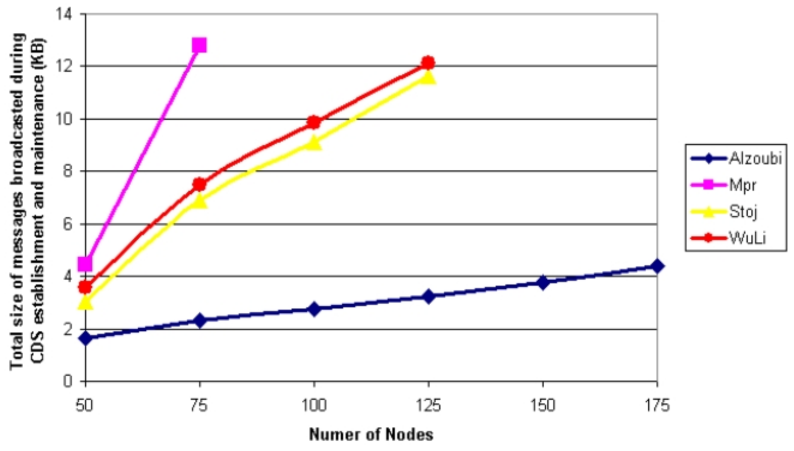

Figure 5.50: The total number of bytes broadcasted at speed range 20-25 m/s.

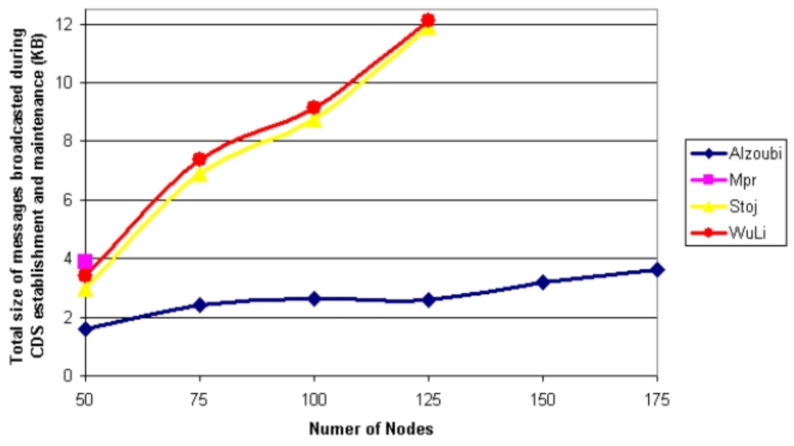

Figure 5.51: The total number of bytes broadcasted at speed range 25-30 m/s.

total number of bytes unicasted by Alzoubi algorithm is the largest among all the studied algorithms.

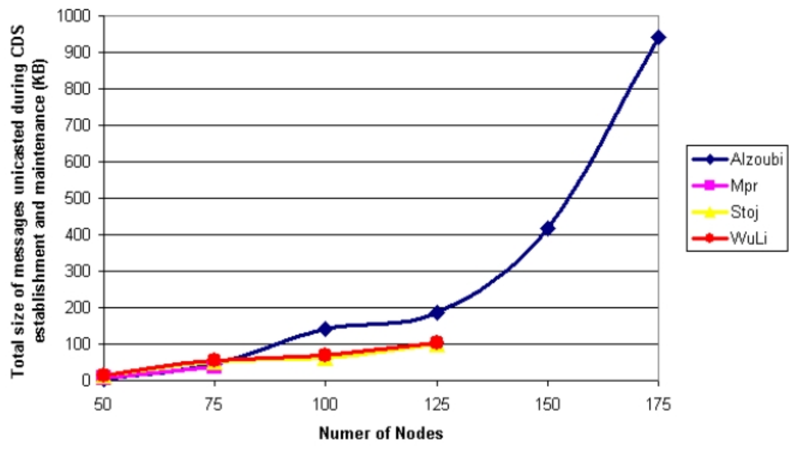

Figure 5.52: The total number of bytes unicasted at speed range 15-20 m/s.

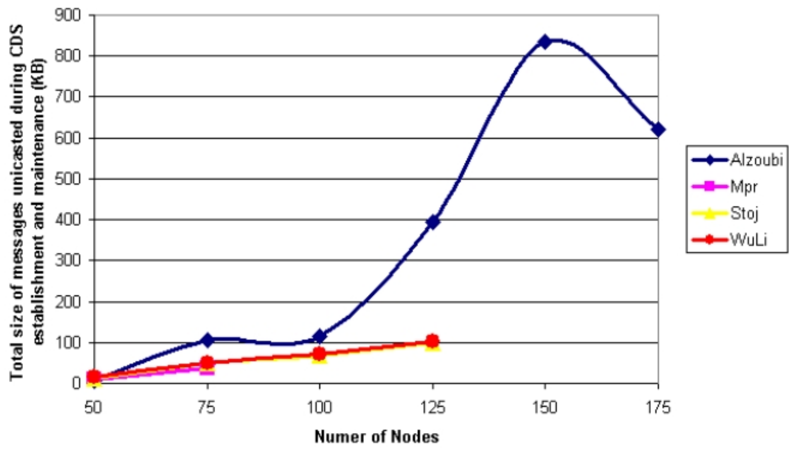

Figure 5.53: The total number of bytes unicasted at speed range 20-25 m/s.

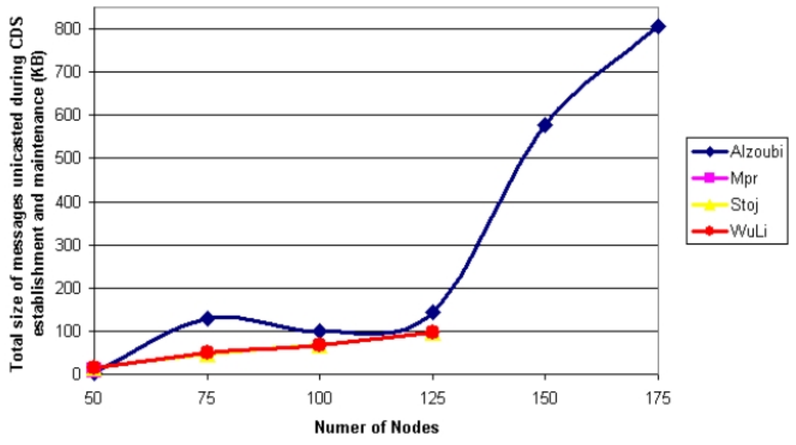

Figure 5.54: The total number of bytes unicasted at speed range 25-30 m/s.

- Results obtained using the *IEEE 802.11 MAC*

Here too, Wu and Li algorithm, Stojmenovic Algorithm, and the MPR algorithm were not able to function properly using the IEEE 802.11 MAC while Alzoubi algorithm show some ability to function under this MAC protocol; however, the performance of Alzoubi algorithm based on the IEEE 802.11 MAC protocol deviates significantly from its performance based on the virtual ideal MAC protocol. The following figures show this performance deviation and the effect of the speed of node movement on it.

- **Deviation in CDS size**

The average size of the CDS produced by Alzoubi algorithm based on the use of the IEEE 802.11 MAC protocol and the corresponding average CDS size produced by the virtual ideal MAC protocol as a function of the number of nodes for three ranges of speed are shown in figures 5.55, 5.56, and 5.57. The important remark to be made about these figures is that the use of the IEEE 802.11 MAC protocol under the Freeway mobility model affects the CDS size in the same way it did under the Random Waypoint mobility model, i.e., it leads to an increase in the CDS size and this size increase becomes more apparent with the increase in the number of nodes.

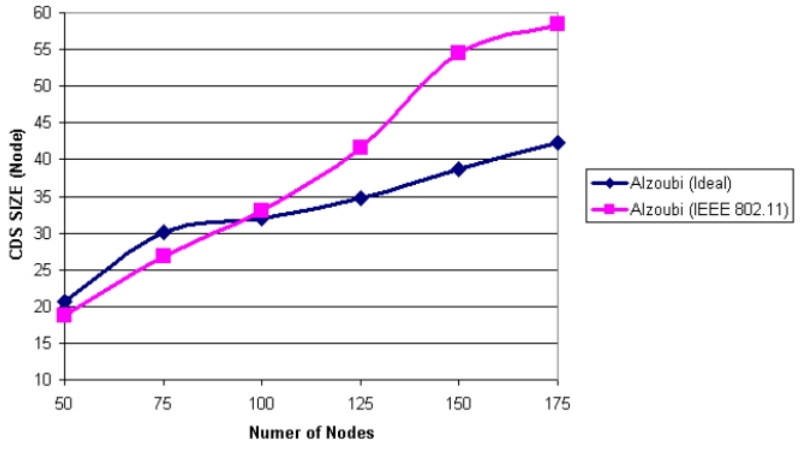

Figure 5.55: Deviation in CDS size at speed range 15-20 m/s.

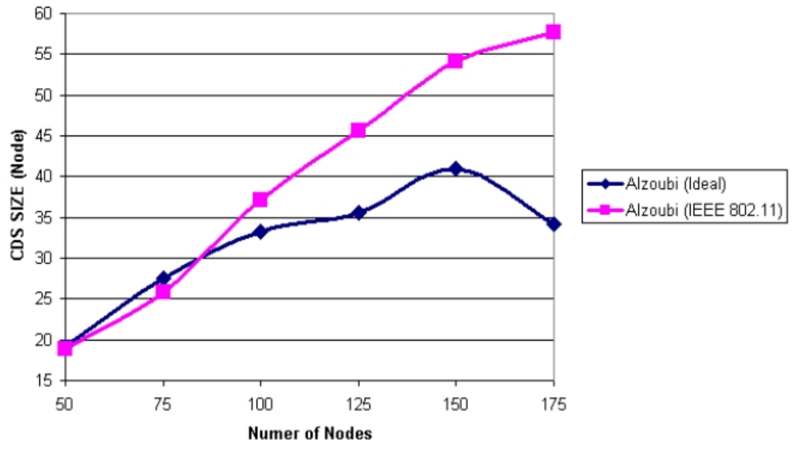

Figure 5.56: Deviation in CDS size at speed range 20-25 m/s.

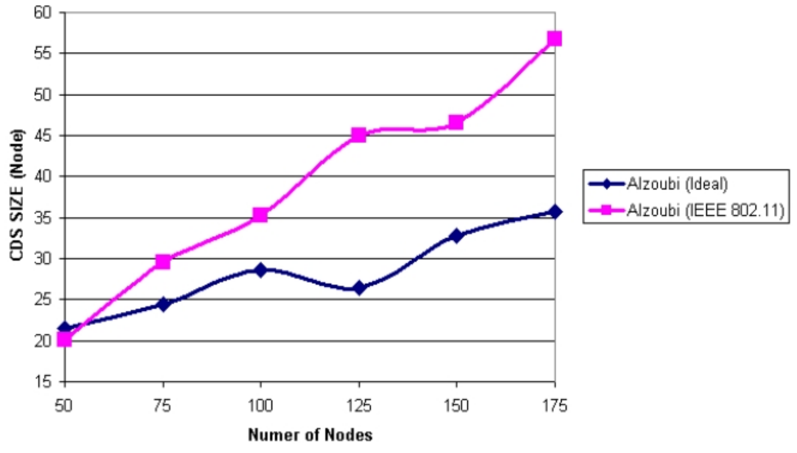

Figure 5.57: Deviation in CDS size at speed range 25-30 m/s.

- **Deviation in CDS establishment time**

Figures 5.58, 5.59, and 5.60 show the effect of the IEEE 802.11 MAC protocol on the CDS establishment time of Alzoubi algorithm. It is apparent in these figures that the CDS establishment time based under the IEEE 802 MAC deviates significantly from that achieved under the virtual ideal MAC protocol. Here too, the deviation increases with the increase in the number of nodes.

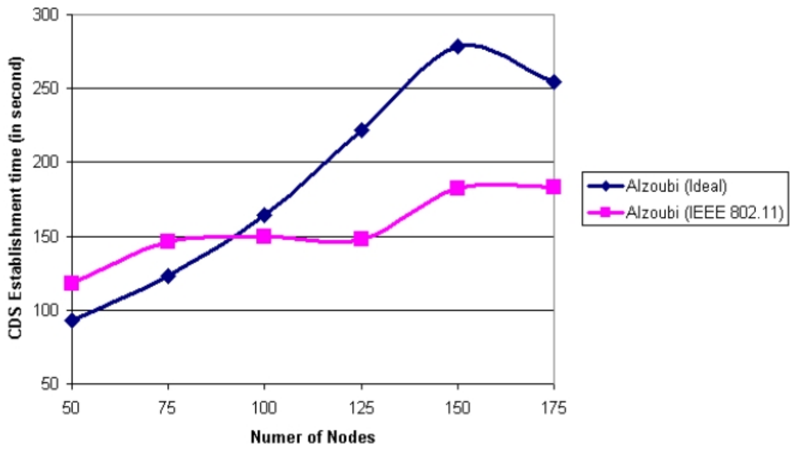

Figure 5.58: Deviation in CDS establishment time at speed range 15-20 m/s.

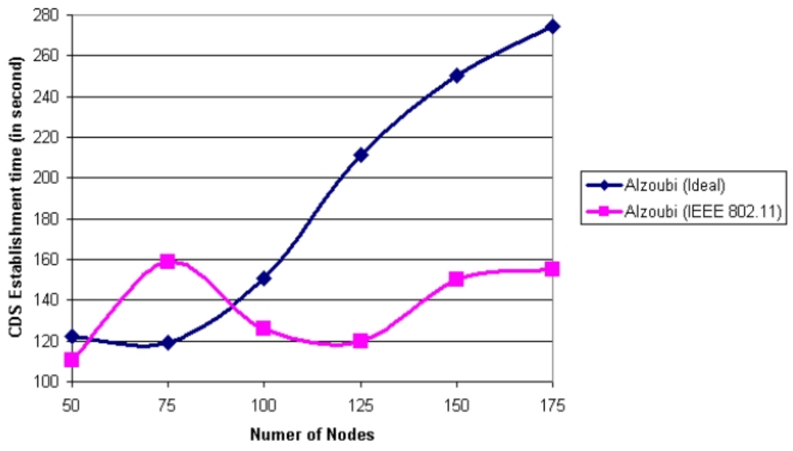

Figure 5.59: Deviation in CDS establishment time at speed range 20-25 m/s.

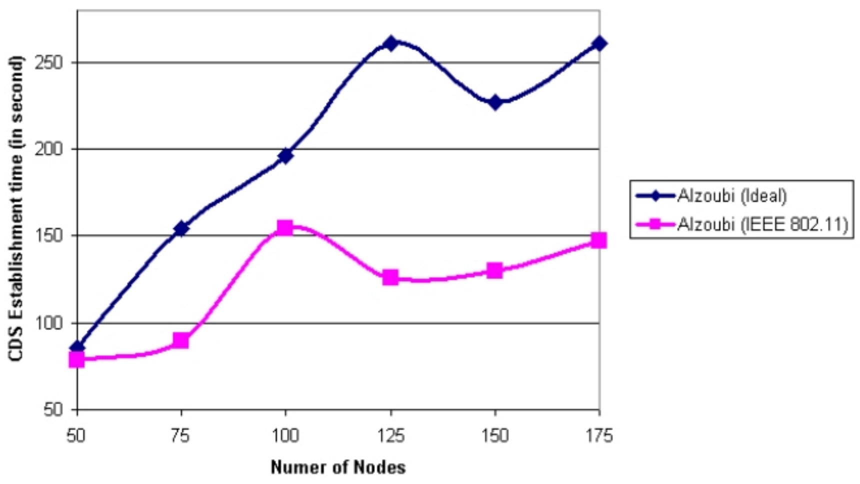

Figure 5.60: Deviation in CDS establishment time at speed range 25-30 m/s.

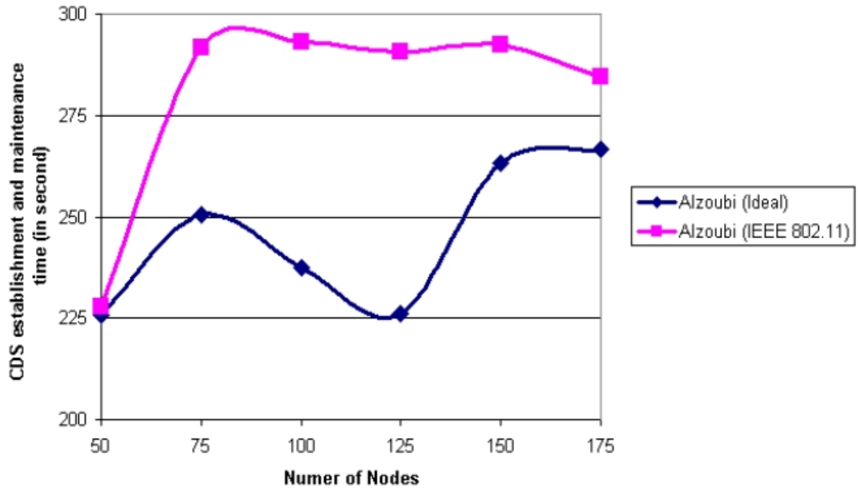

Figure 5.61: The deviation in the total running time at speed range 15-20 m/s.

- **Deviation in the total running time**

The deviation in the total running time as a function of the number of nodes for three ranges of speed are given in figures 5.61, 5.62, and 5.63. It is obvious that the use of IEEE 802.11 MAC protocol leads to longer total running time. Since the CDS establishment time based on the use of the IEEE 802.11 MAC is shorter than that based on the use of the virtual ideal MAC, one can make the same conclusion as that made under the Random Waypoint mobility model, i.e., the CDS maintenance time under the IEEE 802.11 MAC is longer than that under the virtual ideal MAC.

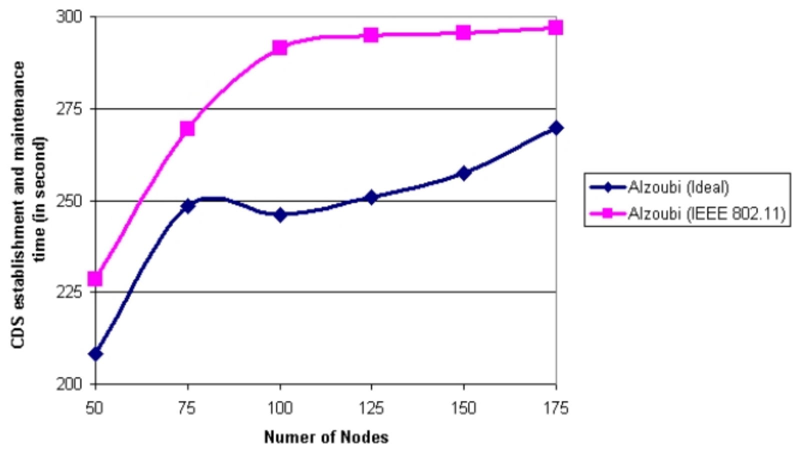

Figure 5.62: The deviation in the total running time at speed range 20-25 m/s.

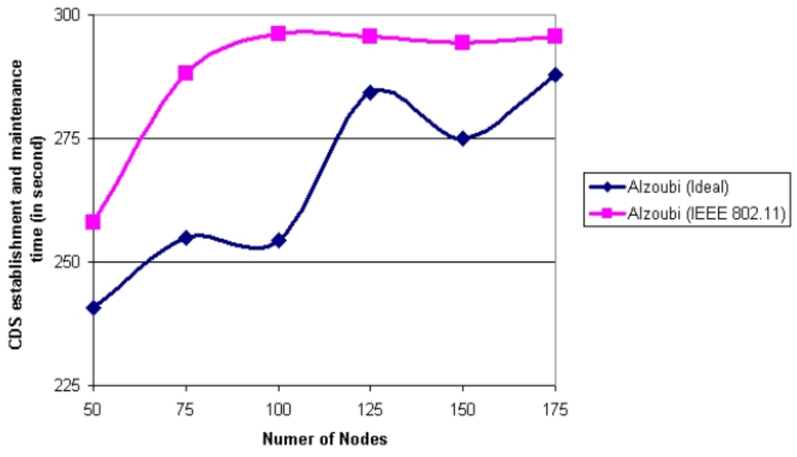

Figure 5.63: The deviation in the total running time at speed range 25-30 m/s.

- **Deviation in the total number of transmitted bytes**

Figures 5.64, 5.65, and 5.66 show the deviation in the total number of transmitted bytes as a function of the number of nodes for three ranges of speed. It is apparent that the bandwidth requirements of Alzoubi algorithm exceeds the bandwidth offered by the wireless communication channel regardless of the mobility model used.

- **Deviation in the total number of broadcasted bytes**

The deviation in the total number of broadcasted bytes is shown in figures 5.67, 5.68, and 5.69. The light use of broadcasting in Alzoubi algorithm is the cause of this small deviation compared to the deviation in the number of transmitted bytes.

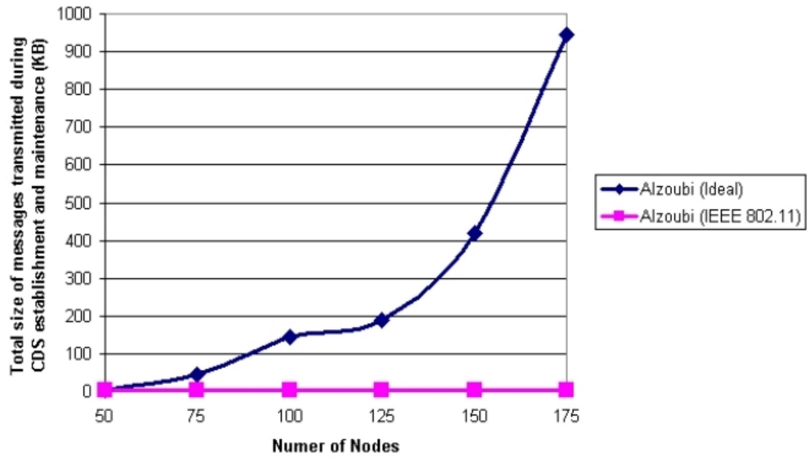

Figure 5.64: The deviation in the number of transmitted bytes at 15-20 m/s.

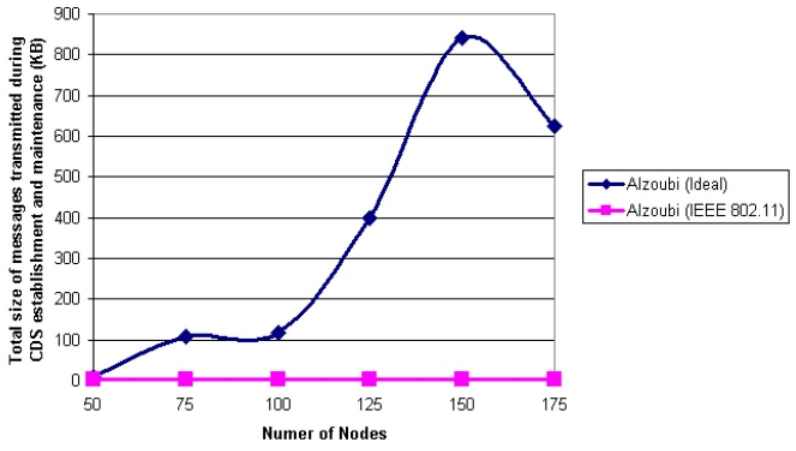

Figure 5.65: The deviation in the number of transmitted bytes at 20-25 m/s.

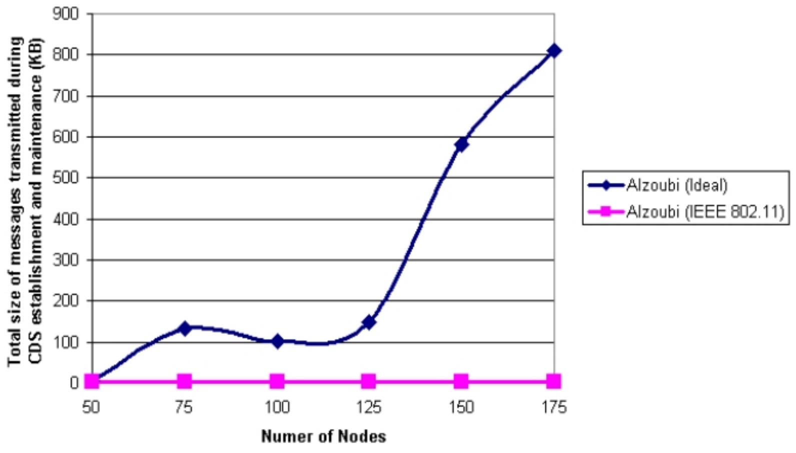

Figure 5.66: The deviation in the number of transmitted bytes at 25-30 m/s.

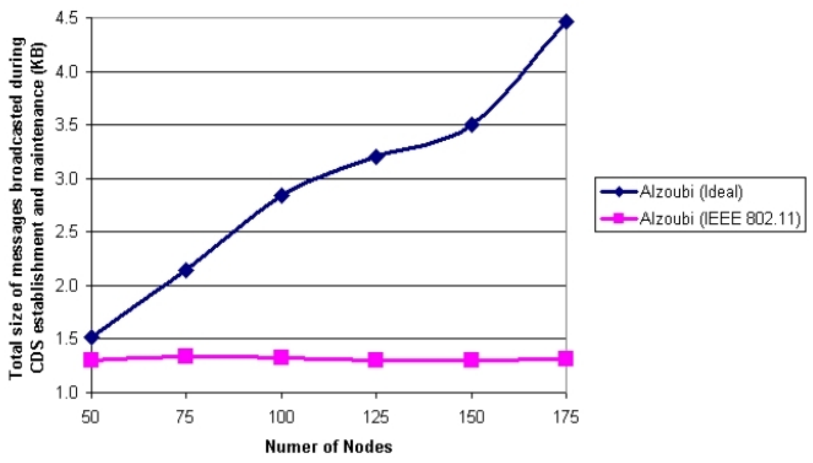

Figure 5.67: The deviation in the number of bytes broadcasted at 15-20 m/s.

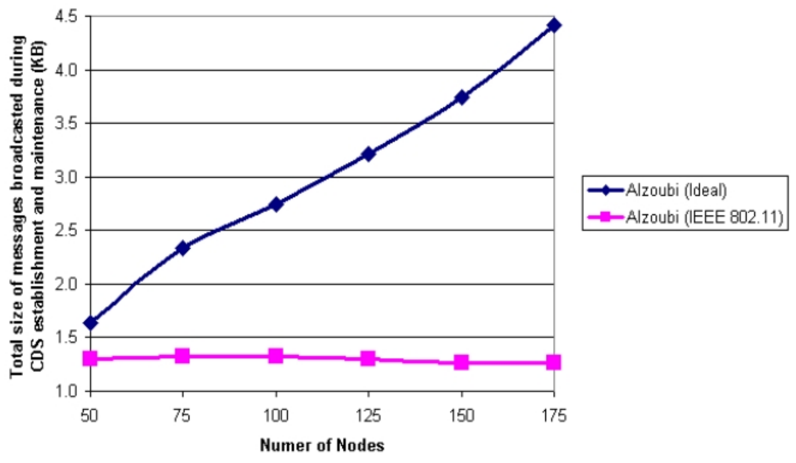

Figure 5.68: The deviation in the number of bytes broadcasted at 20-25 m/s.

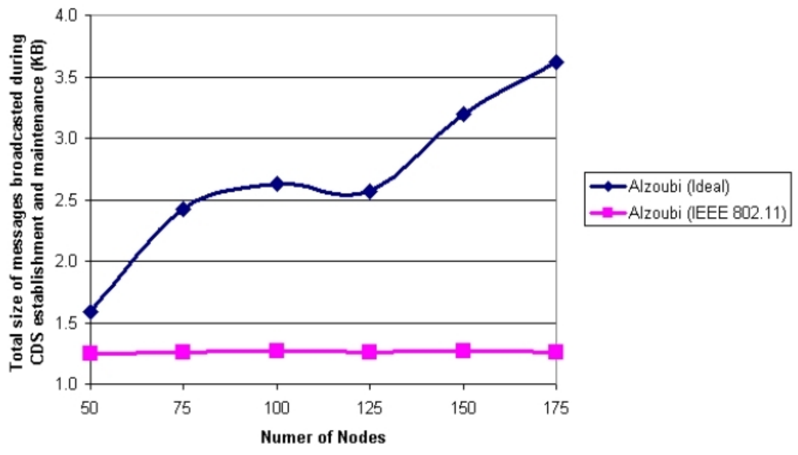

Figure 5.69: The deviation in the number of bytes broadcasted at 25-30 m/s.

- **Deviation in the total number of unicasted bytes**

Figures 5.70, 5.71, and 5.72 show the deviation in the total number of unicasted bytes as a function of the number of nodes for three ranges of speed. Due to the heavy use of unicasting in Alzoubi algorithm, it can be noticed that the deviation in the total number of unicasted bytes is much larger than the deviation in total number of broadcasted bytes.

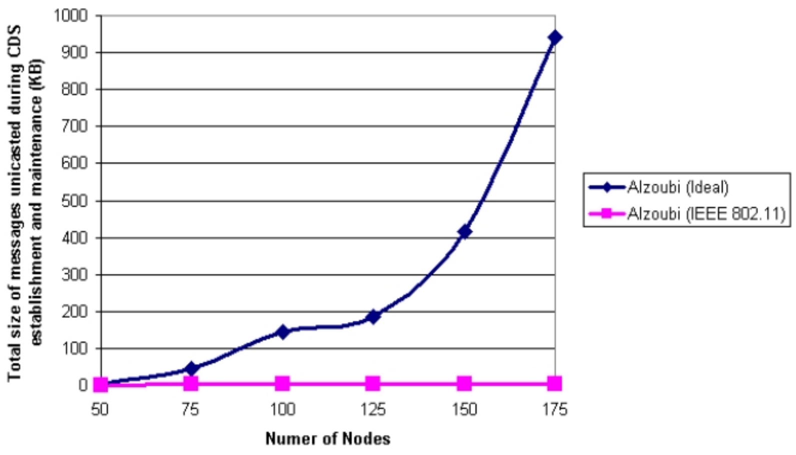

Figure 5.70: The deviation in the number of bytes unicasted at 15-20 m/s.

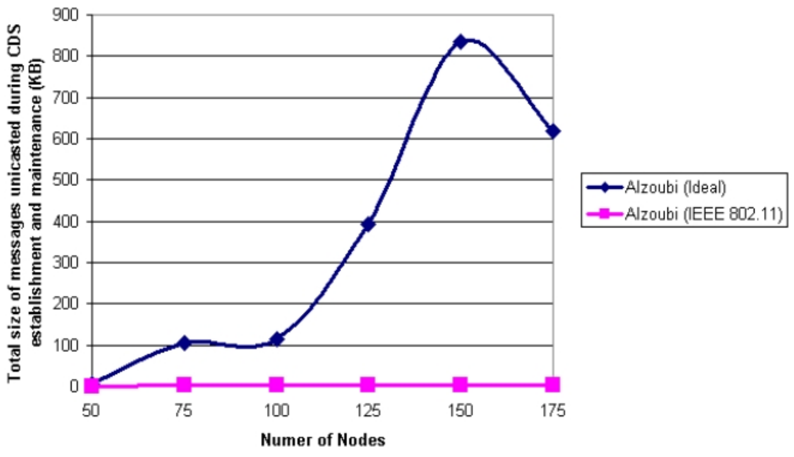

Figure 5.71: The deviation in the number of bytes unicasted at 20-25 m/s.

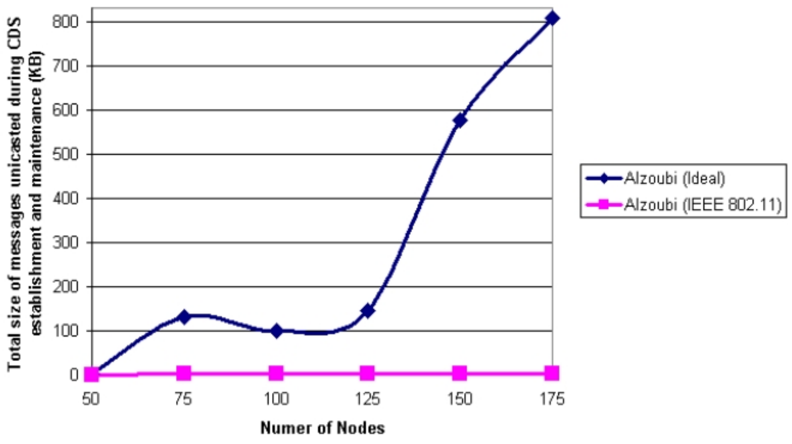

Figure 5.72: The deviation in the number of bytes unicasted at 25-30 m/s.

Chapter 6

Conclusions and Future Work

In this thesis, the performance of some of the most renowned virtual backbone formation algorithms, namely: Alzoubi algorithm, Wu and Li algorithm, Stojmenovic algorithm, and the MPR algorithm, is evaluated in a simulation-based experiment. This chapter presents the conclusions drawn from the outcomes of the evaluation of the aforementioned algorithms.

6.1 Conclusions

Based on the results presented in the previous chapter and under the same conditions pertaining to the number of nodes and the speed of movement as those used in the simulations described in this thesis, the following conclusions regarding the real life behavior of the studied algorithms are drawn:

- The narrow bandwidth of the wireless communications channel renders all of the algorithms studied in this thesis, except Alzoubi algorithm, useless since they fail in constructing virtual backbones using the IEEE 802.11 MAC protocol.
- Among the algorithms studied in this thesis, Alzoubi algorithm is the only algorithm that showed some ability to cope with the restrictions imposed by the bandwidth limitations; however, the number of nodes that can be accommodated by this algorithm has an upper limit of 50 nodes.
- The mobility models used in the simulations described in this thesis, namely: the Randomway Point mobility model and the Freeway mobility model, have

no impact on the ranking of the studied algorithms based on any of the performance measures employed.

However, ignoring the restrictions imposed by the channel bandwidth and granting the design assumptions of the studied algorithms of guaranteed delivery of all messages lead to the following conclusions:

- Alzoubi algorithm is the most scalable algorithm and the MPR algorithm is the least. The scalability of Wu and Li algorithm and Stojmenovic algorithm are the same.
- The fastest algorithm in the CDS creation phase is Stojmenovic algorithm while the MPR algorithm is the slowest. The CDS creation time of Alzoubi algorithm and Wu and Li algorithm are comparable.
- In general, Alzoubi algorithm is the fastest algorithm in CDS maintenance; however, in MANETs that have 70 nodes or less, Stojmenovic algorithm outperforms Alzoubi algorithm in this regard.

item Alzoubi algorithm has the highest signalling overhead among all the studied algorithms; however, most of this overhead is sent by unicasting and the portion sent by broadcasting is the lowest compared to others. As a result, Alzoubi algorithm does not suffer from sever message loss due to collisions, whereas almost all of the signalling overhead of other algorithms is sent by broadcasting.

6.2 Future Work

Virtual backbone is proposed as a solution to the problem of routing in MANETs. This thesis dealt with the quality of the virtual backbones themselves, and the next step should investigate the ability of these virtual backbones to solve the problem of routing in MANETs.

Another possible extension to this thesis is to go a further step in the direction of testing the quality of the virtual backbones and to evaluate their routing performance. Even though the simulation-based performance evaluation provides a good insight about the real life performance of the studied algorithms, it does not eliminate the need for the real life testing for these algorithms. The real life implementation and testing of the algorithms studied in this thesis is another recommended future step.

List of References

[1] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. : A review of routing protocols for mobile ad hoc networks. *Ad hoc Networks*, 2(1):1–22, 2004. 3

[2] C. Adjih, P. Jacquet, and L. Viennot. : Computing connected dominated sets with multipoint relays. 4, 53, 67, 68

[3] G. Aggelou and R. Tafazolli. : A bandwidth-efficient routing protocol for mobile ad hoc networks. pages 26–33, 1999. in: ACM International Workshop on Wireless Mobile Multimedia (WoWMoM). 29

[4] K. M. Alzoubi. *Virtual Backbones in Wireless Ad Hoc Networks*. PhD thesis, Illinois Institute of Technology, 2002. 69, 82

[5] K. M. Alzoubi, P.-J. Wan, and O. Frieder. : Distributed heuristics for connected dominating sets in wireless ad hoc networks. *Journal of Communications and Networks*, 4(1):22–29, 2002. 54, 55, 56

[6] K. M. Alzoubi, P.-J. Wan, and O. Frieder. : Message-optimal connected-dominating-set construction for routing in mobile ad hoc networks. 2002. in 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc02). 3, 4, 55, 57, 62, 63, 64, 69

[7] K. M. Alzoubi, P.-J. Wan, and O. Frieder. : New distributed algorithm for connected dominating set in wireless ad hoc networks. Jannuary 2002. HICSS35. 4, 54, 55, 56, 61, 69

[8] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Weakly-connected dominating sets and sparse spanners in wireless ad hoc networks. In *ICDCS*, pages 96–104, 2003. 61

[9] A. Amis, R. Prakash, T. Vuong, and D. Huynh. : Max-min d-cluster formation in wireless ad hoc networks. Tel Aviv, March 2000. in Proceedings of in Proceedings of IEEE INFOCOM. 50

[10] B. An and S. Papavassiliou. : A mobility-based clustering approach to support mobility management and multicast routing in mobile ad hoc wireless networks. *International Journal of Network Management*, pages 387–395, 2001. 49

[11] F. Bai, S. Narayanan, and A. Helmy. : Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks. pages 825–835, San Francisco., April 2003. IEEE INFOCOM (The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies). 85

[12] D.J. Baker and A. Ephremides. : The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981. 48

[13] S. Basagni. : Distributed clustering for ad hoc networks. pages 310–315, 1999. in Proceedings of ISPAN'99 Int. Symp. on Parallel Architectures, Algorithms, and Networks.

[14] S. Basagni, I. Chlamtac, V.R. Syrotivk, and B.A. Woodward. : A distance effect algorithm for mobility (dream). Dallas, TX, 1998. in: Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking Mobicom 98. 11

[15] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli. : Localized protocols for ad hoc clustering and backbone formation: A performance comparison. *IEEE Trans. Parallel Distrib. Syst*, 17(4):292–306, 2006. 45, 48, 52, 53, 63, 68

[16] P. Basu, N. Khan, , and T. D. C. Little. : A mobility based metric for clustering in mobile ad hoc networks. pages 413–418, April 2001. in Proceedings IEEE ICDCSW01. 48

[17] E. M. Belding-Royer. : Multi-level hierarchies for scalable ad hoc routing. *Wireless Networks*, 9:461–478, 2003. 51

[18] B. Bellur, R. G. Ogier, and F. L. Templin. : topology broadcast based on reverse-path forwarding routing protocol (tbrpf). 2003. in : Internet Draft, draft-ietf-manet-tbrpf-06.txt, work in progress. 18

[19] S. Butenko, X. Cheng, D.-Z. Du, and P.M. Pardalos. *Cooperative Control: Models, Applications and Algorithms*, chapter On The Construction of Virtual

Backbone for Ad Hoc Wireless Networks, pages 43–54. Kluwer Academic Publishers, 2003. 59

[20] S. Butenko, X. Cheng, C. Oliveira, and P.M. Pardalos. *Recent Developments in Cooperative Control and Optimization*, chapter A New Heuristic for The Minimum Connected Dominating Set Problem on Ad Hoc Wireless Networks, pages 61–73. Kluwer Academic Publishers, 2004. 59

[21] M. Chatterjee, S.K. Das, and D. Turgut. : Wca: A weighted clustering algorithm for mobile ad hoc networks. *Cluster Computing*, 5(2):193–204, 2002. 49

[22] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. : An energy-effecient coordination algorithm for topology maintenance in ad hoc wireless networks. pages 85–96. ACM Press, 2001. In Proceedings of the 7th annual international conference on Mobile computing and networking. 53

[23] G. Chen, F. Nocetti, J. Gonzalez, and I. Stojmenovic. : Connectivity-based k-hop clustering in wireless networks. January 2002. in Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS-35). 48

[24] T. W. Chen and M. Gerla. : Global state routing: A new routing scheme for ad hoc wireless networks. 1998. in Proceedings of the the IEEE ICC. 9

[25] Y. P. Chen and A. L. Liestman. : A zonal algorithm for clustering ad hoc networks. *International Journal of Foundations of Computer Science*, 14(2):305–322, 2003. 61

[26] Y. P. Chen and A. L. Liestman. : Maintaining weakly-connected dominating sets for clustering ad hoc networks. *Ad Hoc Networks*, 3(5):629–642, September 2005. 61

[27] Y. P. Chen, A. L. Liestman, and J. Liu. *Ad Hoc and Sensor Networks*, chapter Clustering Algorithms for Ad Hoc Wireless Networks, pages 400–416. Nova Science Publishers, 2004. 47

[28] Yuanzhu Peter Chen and Arthur L. Liestman. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *MobiHoc*, pages 165–172, 2002. 60, 61

[29] Xiuzhen Cheng and Ding-Zhu Du. : Virtual backbone-based routing in multihop ad hoc wireless networks. University of Minnesota, June 2002. Technical Report 002. 59

[30] C. C. Chiang. : routing in clustered multihop mobile wireless networks with fading channel. pages 197–211, April 1997. in: Proceedings of IEEE SICON. 14

[31] C. C. Chiang, H. K. Wu, W. Liu, and M. Gerla. : Routing in clustered multihop mobile wireless networks with fading channel. pages 197–211, April 1997. IEEE Singapore International Conference on Networks(SICON). 48

[32] Z. Chun-xiao and W. Guang-xing. : Fuzzy-control-based clustering strategy in manets. volume 2, pages 1456–1460, September 2000. in Proceedings of the fifth world congress on Intelligent control and automation. WCICA. 49

[33] I. Cidon and O. Mokryn. : Propagation and leader election in multihop broadcast environment. pages 104–119, September 1998. in Proceedings of 12th Int. Symp. Distr. Computing. 54, 55

[34] M. S. Corson and A. Ephremides. : A distributed routing algorithm for mobile wireless networks. *ACM Journal on Wireless Networks*, 1:61–81, Feb. 1995. 22, 23

[35] V. D. and M. S. Corson Park. : A highly adaptive distributed routing algorithm for mobile wireless networks. pages 522–527, April 1997. in: Proceedings of INFOCOM. 23

[36] F. Dai and J. Wu. : An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Trans. Parallel and Distributed Systems*, 15(10):908–920, 2004. 65

[37] B. Das and V. Bharghavan. : Routing in ad hoc networks using minimum connected dominating sets. Montreal, Canada, June 1997. International Conference on Communications. 52

[38] S. Das, C. Perkins, and E. Royer. : ad hoc on demand distance vector (aodv) routing. 2002. in : Internet Draft, draft-ietf-manetaodv- 1.txt, work in progress. 20, 32

[39] M. Dorigo and G. Di Caro. : The ant colony optimization meta-heuristic. 1999. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 11-32. McGraw-Hill, London. 32

[40] R. Dube, C. Rais, K. Wang, and S. Tripathi. : Signal stability based adaptive routing (ssa) for ad hoc mobile networks. *IEEE Transactions on Communications*, 4(1):36–45, 1997. 28

[41] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. : Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. pages 717–724, 2003. in Proc. ACM- SIAM Symposium on Discrete Algorithms (SODA). 53, 54

[42] I. I. Er and K. G. Winston Seah. : Mobility-based d-hop clustering algorithm for mobile ad hoc networks. Atlanta, Georgia, USA, March 21-25, 2004. Proceedings of IEEE Wireless Communications and Networking Conference. 45, 49

[43] J. eremy, B. M. Ding, A. Thaeler, and X. Cheng. *Handbook of Combinatorial Optimization*, chapter Connected Dominating Set in Sensor Networks and MANETs, pages 329–369. Kluwer Academic Publishers, 2004. 51

[44] K. Fall and K. Varadhan. *The ns Manual (formerly ns Notes and Documentation).* The VINT Project. 84

[45] E. Gafni and D. Bertsekas. : Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 29(1):11–18, January 1981. 23

[46] R. Gandhi, S. Parthasarathy, and A. Mishra. : Minimizing broadcast latency and redundancy in ad hoc networks. pages 222–232. ACM Press, 2003. In Proceedings of the fourth ACM international symposium on Mobile ad hoc networking and computing. 58

[47] J.J. Garcia-Luna-Aceves and C. Marcelo Spohn. : Source-tree routing in wireless networks. New Orleans, 1999. in: Proceedings of the Seventh Annual International Conference on Network Protocols Proceedings of Wireless Communications and Networking. 10

[48] M. Gerla, T. J. Kwon, and G. Pei. : On demand routing in large ad hoc wireless networks with passive clustering. September 2000. in Proceedings of IEEE WCNC. 50

[49] M. Gerla and J. T. Tsai. : Multiuser, mobile, multimedia radio network. *Wireless Networks*, 1:255–65, October 1995. 48

[50] S. Guha and S. Khuller. : Approximation algorithms for connected dominating sets. *Algorithmica*, 20:374–387, 1998. 4, 52, 60

[51] P. Gupta, R. Gray, and P. Kumar. : An experimental scaling law for ad hoc networks. 3

[52] M. Guunes, U. Sorges, and I. Bouazizi. : Ara the ant-colony based routing algorithm for manets. pages 79–85, August 2002. in: ICPP workshop on Ad Hoc Networks (IWAHN 2002). 32

[53] Z. J. Hass and R. Pearlman. : zone routing protocol for ad hoc networks. 1999. in : Internet Draft, draft-ietf-manet-zrp-02.txt, work in progress. 36

[54] T. C. Hou and V. O. K. Li. : Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1), January 1986. 38

[55] http://grouper.ieee.org/groups/802/11/main.html. 84, 95

[56] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. : Optimized link state routing protocol for ad hoc networks. Pakistan, 2001. IEEE INMIC. 17

[57] L. Jia, R. Rajaraman, and T. Suel. : An effecient distributed algorithm for constructing small dominating sets. 2001. in Proceedings of the 20th ACM Symposium on Principles of Distributed Computing (PODC'01). 51, 53

[58] M. Jiang, J. Ji, and Y. C. Tay. : cluster based routing protocol. 1999. in : Internet Draft, draft-ietf-manet-cbrp-spec-01.txt, work in progress. 35

[59] M. Joa-Ng and I. T. Lu. : A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1415–1425, 1999. 37

[60] T. Johansson and L. Carr-Motyckova. : Bandwidth-constrained clustering in ad hoc networks. 50

[61] D. Johnson, D. Maltz, and J. Jetcheva. : the dynamic source routing protocol for mobile ad hoc networks. 2002. in : Internet Draft, draft-ietf-manet-dsr-07.txt, work in progress. 19, 26, 34, 85

[62] K.K. Kasera and R. Ramanathan. : A location management protocol for hierarchically organised multihop mobile wireless networks. pages 158–162, San Diego, CA, October 1997. in: Proceedings of the IEEE ICUPC 97. 12

[63] Y. B. Ko and N. H. Vaidya. : Location-aided routing (lar) in mobile ad hoc networks. Dallas, XT, 1998. in: Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 98). 30

[64] B. Liang and Z. J. Haas. : Virtual backbone generation and maintenance in ad hoc network mobility management. pages 1293–1302, 2000. in INFOCOM. 48, 51

[65] C. R. Lin and M. Gerla. : Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, 1997. 45

[66] M. V. Marathe, H. Breu, H. B. Hunt, S. S. Ravi, and D. J. Rosenkrantz. : Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995. 52

[67] J. Matousek. *Lectures on Discrete Geometry. Graduate Texts in Mathematics.* Springer, New York, 2002. 54

[68] A. B. McDonald and T. Znati. : A mobility based framework for adaptive clustering in wireless ad-hoc networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1466–1487, August 1999. 45, 62

[69] A. B. McDonald and T. Znati. : Predicting node proximity in ad-hoc networks: A least overhead adaptive model for selecting stable routes. volume 17, pages 1466–1487, August 1999. 62

[70] A. B. McDonald and T. Znati. : Statistical estimation of link availability and its impact on routing in wireless ad hoc networks. *Wireless Communications and Mobile Computing*, 4(4):331–349, June 2004. 62

[71] A. B. McDonald and T. Znati. : A path availability model for wireless ad-hoc networks. New Orleans, LA, Septmber 1999. in Proceedings of the IEEE Wireless Communications and Networking Conference 1999 (WCNC'99). 62

[72] S. Murthy and J. J. Garcia-Luna-Aceves. : A routing protocol for packet radio networks. pages 86–95, Berkeley, CA, 1995. in Proceedings of the First Annual ACM International Conference on Mobile Computing and Networking. 7

[73] N. Nikaein, C. Bonnet, and N. Nikaein. : Harp: Hbrid ad hoc routing protocol. Tehran, Iran, September 1-3 2001. in: Proceedings of IST: International Symposium on Telecommunications. 42

[74] N. Nikaein, H. Laboid, and C. Bonnet. :distributed dynamic routing algorithm (ddr) for mobile ad hoc networks. 2000. in: Proceedings of the MobiHOC 2000: First Annual Workshop on Mobile Ad Hoc Networking and Computing. 41, 42

[75] S. Parthasarathy and R. Gandhi. : Fast distributed well connected dominating sets for ad hoc networks. University of Maryland, Computer Science Department, 2004. Technical Report CS-TR-4559. 57, 58

[76] G. Pei, M. Gerla, and T. W. Chen. : Fisheye state routing: A routing scheme for wireless ad hoc networks. New Orleans, LA, june 2000. Proc ICC 2000. 9

[77] G. Pei, M. Gerla, X. Hong, and C. Chiang. : A wireless hierarchical routing protocol with group mobility. pages 158–162, New Orleans, 1999. in: Proceedings of Wireless Communications and Networking. 15

[78] C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001. 1

[79] C. E. Perkins and T.J. Watson. : Highly dynamic destination sequenced distance vector routing dsdv for mobile computers. London, UK, 1994. ACM SIGCOMM 94 Conference on Communications Architectures. 7

[80] A. Qayyum, L. Viennot, and A. Laouiti. : Multipoint relaying for flooding broadcast message in mobile wireless networks. pages 3898–3907, January 2002. in Proc. of 35th Hawaii Intl Conf. on System Sciences (HICSS-35). 53, 67, 68

[81] S. Radhakrishnan, N. S. V Rao, G. Racherla, C. N. Sekharan, and S. G. Batsell. :dst a routing protocol for ad hoc networks using distributed spanning trees. New Orleans, 1999. in: IEEE Wireless Communications and Networking Conference. 40

[82] Raju and J. Garcia-Luna-Aceves. : a new approach to on demand loop-free multipath routing. pages 522–527, Boston, MA, October 1999. in: Proceedings of the 8th Annual IEEE International Conference on Computer Communications and Networks (ICCCN). 21

[83] C. Santivanez, R. Ramanathan, and L. Stavrakakis. : Making link-state routing scale for ad hoc networks). Long Beach, California, October 2001. in: 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHOC 2001. 3

[84] R. Sivakumar, P. Sinha, and V. Bharghavan. : Cedar: A core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications*, 17(8):1454–1465, Augaust 1999. 50

[85] I. Stojmenovic, M. Seddigh, and J. Zunic. : Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Trans. Parallel and Distributed Systems*, 13:14–25, 2002. 4, 53, 66

[86] W. Su and M. Gerla. : Ipv6 flow handoff in ad hoc wireless networks using mobility prediction. pages 271–275, Rio de Janeiro, Brazil, December 1999. in: IEEE Global Communications Conference. 34

[87] C. Toh. : A novel distributed routing protocol to support ad hoc mobile computing. pages 480–486, 1996. in: IEEE 15th Annual International Phoenix Conf. 26

[88] N. H. Vaidya, P. Krishna, M. Chatterjee, and D. K. Pradhan. : A cluster-based approach for routing in dynamic networks. *ACM Computer Communications Review*, 27(2), 1997. 45, 62

[89] P. Wan, K. Alzoubi, and O. Frieder. : distributed construction of connected dominating set in wireless ad hoc networks. *Mobile Networks and Applications*, 9(2):141–149, April 2004. 3, 48, 55, 56, 57, 64

[90] S. C. Woo and S. Singh. : scalable routing protocol for ad hoc networks. *Wireless Networks*, 7(5):513–529, 2001. 38

[91] J. Wu. : An enhanced approach to determine a small forward node set based on multipoint relays. volume 4, pages 2774–2777, September 2003. in Proc. of IEEE VTC 2003 fall. 53, 68

[92] J. Wu and H. Li. : On calculating connected dominating set for effeicient routing in ad hoc wireless networks. pages 7–14, Seattle, 1999. in DIAL M'99. 3, 4, 52, 53, 65

[93] J. Wu, W. Lou, and F. Dai. : Extended multipoint relays to determine connected dominating sets in manet. *IEEE Trans. on Computers*, 55(10):334–347, March 2006. 53

[94] X. Y. Xu and M. Gerla. : Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, 16(4):11–21, July-Aug. 2002.

[95] Y. Yi, M. Gerla, and T. Kwon. : Efficient flooding in ad hoc networks using on-demand (passive) cluster formation. in MOBIHOC 2002 Poster Session. 49

[96] J. Y. YU and P. H. J. CHONG. : A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7(1):32–48, First Quarter 2005. 63