Universidad EAFIT

Departamento de Informática y Sistemas

Nodes Selection Algorithm for the Implementation of an Intrusion Detection System in a Mobile Ad Hoc Network

---

A thesis submitted to the faculty of
EAFIT UNIVERSITY

in partial fulfillment of the requirements
for the degree of Master in Engineering - Computer Science

Author:
Diana Patricia Vélez Granada

Thesis advisor:
Prof. PhD. Juan Guillermo Lalinde Pulido

Medellín , Colombia
2010

To my family, who supported me, and Edgar, who has given meaning to everything I do.

Acknowledments

It is not possible for me to say thanks without thinking about God, who gives me the life and protects me even when I forget him.
Thanks God.

I would like specially thank to my family, Camilo, Noralba and Sandra, for their support and faith in me.
I could not have been what I am without you.

Thanks to Edgar, my boyfriend, for being a good example to follow, for being my shoulder and hear me at all times.
Thanks you for your love.

Thanks to Juan Guillermo, my advisor, for his help, patience and jokes.
Working with you has been an enriching experience.

I will not forget the teachers who contributed in my academic and personal education.
Thanks to each one of you.

I also thank each of my friends and colleagues, especially those who always asked me about my thesis.
The generated pressure has borne fruit today.

Diana Patricia Vélez Granada

Contents

List of Figures

List of Tables

Introduction

Literature about Intrusion Detection Systems (IDS), shows their tasks have been studied enough and are clearly defined to monitor and analyze events, looking for evidence that indicate the presence of intruders[12]. When the purpose is to implement an IDS in a Mobile Ad hoc Network (MANET), the first things that are advertised are the lack of a defense line and physical borders and the limited resources of the devices. So one important question is: where can an IDS processing node be located in a MANET?.

IDS are expensive in terms of processes, and the independent implementation on a MANET device implies the consumption of an important amount of resources and does not allow global analysis[8]. For this reason the distributed IDS architectures for MANET was proposed. A lot of work has been done about distributed IDS architectures for MANETs, an some go deeper in the analysis of the network in order to identify the nodes that are critical for keeping the network connected[46]. Others suggest practical monitoring procedures that grant host and network events analysis [60]. The strategies of distributed IDS usually consist in distributing IDS's sensors, agents or processes through the network, the events are analyzed locally and then sent to a node (or group of distributed nodes) to be correlated to infer the security status of the entire network [8, 53].

In [60], the authors proposed a monitoring procedure where some nodes take their local traces and send them to a global observer. The global observer must correlate the events in a synchronous way. They do not suggest a location for the sensor nor the global observer. In this work, we propose a strategic location for a distributed IDS of this class, stating the location of both of them, the local sensors (independent IDS, agents or IDS processes) and the global observer.

Due to the dynamic of the network and free movement of nodes, MANETs are a special case for security, and location of IDS sensors and processes must be carefully studied to guarantee the complete or almost complete coverage of the network. Our strategy consist in locate the process for the correlation of events in a region of the network that minimize the number of hops of the messages that are sent from the sensors, and to maximize the stability of the IDS through the spread of the sensors in the network in a way that allows good coverage and analysis of the events.

Chapter 1

Mobile Ad Hoc Networks

A Mobile Ad Hoc Network (MANET) is a network composed by two or more mobile nodes that communicate through wireless links. These kind of networks have becoming an important research field in the last years[4, 87] due to the large number of applications and easy deployment due to the lack of a fixed infrastructure.

Research in mobile ad hoc networks is quite wide, going from the physical layer to the application layer[105]. Big challenges are imposed not only by the few resources present in a mobile node but also by the dynamic of the topology and the open communication medium.

This chapter shows important concepts of MANETs, in order to get a better understanding of the base of this research.

1.1 Mobile Ad Hoc Networks Features

According to literature[8, 10, 57, 89], a MANET is a self-organized network composed by a set of autonomous wireless mobile nodes that do not depend on a fixed infrastructure or central administration to be able to communicate. This independence is one of the most important and challenging characteristic of the mobile ad hoc networks, implying that the network can be rapidly deployed with low cost[9, 65, 110], and all process in the network must work in a decentralized way including the administration[89].

The decentralized operation means that the processes executed in the network require the cooperation of nodes. As a consequence, the network has low probability to collapse even if a node or multiple nodes are lost, turned off or move away, thus lacking of a unique point of failure, and characterizing this network like a fault tolerant network.

Each node or device has a transmitter and a receiver to establish wireless links and communicate directly with other nodes in range, but if the destination node is outside the coverage it will require the use of intermediate nodes to send messages, procedure known as multi-hop routing. Thus a node should act as a source, destination and intermediate

router[9, 53]. This is why MANETs are sometimes known as "multi-hop wireless ad hoc networks"[57].

In MANETs nodes have the ability to move freely. Therefore they can be characterized by its dynamic topology with unpredictable route changes and frequent network partitions[57]. Additionally, in most cases, nodes are required to have a light weight and be portable, so resources like storage, memory, energy and processing capacity are very limited[10, 89].

The transmission medium in a mobile ad hoc network is as shared as open[57], so signals are susceptible to interfere with other signals that travel through the air. Due to this reason and the network dynamics, the probability collision and loss of packets is increased and the bandwidth of the wireless links is decreased[33].

Usually all the nodes of a mobile ad hoc network are equal[33], and have the same capabilities and responsibilities. However, a MANET can include a mixture of mobile communication devices like personal digital assistants, embedded processors in special purpose devices, mobile phones, laptop computers, tablet computers, etc.[64, 84]. Thus, each node may have different capabilities, due to the diverse types and number of interfaces to transmit and receive that it could have, resulting in asymmetric links and different bandwidths among the network[57, 65, 89].

These network and node characteristics, and the wireless traditional problems, impose certain challenges to the developments in this field.

1.2 Applications

The mobility of nodes, self-configuration and independence from a fixed infrastructure, converts MANETs into a scalable and good solution to be used for unexpected events, which require rapid network establishment and where is no possible to install all the traditional infrastructure.

Nowadays, almost each person has a personal device with some or several wireless technologies[75]. When several people are in a limited region, is more logical to communicate directly using the wireless devices than using an infrastructure based network. This is the key of MANETs: to allow people to communicate using their devices without using a remote and maybe expensive infrastructure.

Although mobile ad hoc networks were developed with military objectives, these networks currently have a wide scope of applications. Examples of applications include monitoring real time data using sensor networks, supporting law enforcement or police operations in special events, supporting rescue missions during natural disasters, transmitting road conditions and weather information using vehicular networks, enhancement of the communication during classes and conferences, responding to interaction rise in multiuser applications and accessing location aware services, among other applications[33, 57].

1.3 Architecture of Mobile Ad Hoc Networks

The protocol stack of a network is the group of protocols that defines the rules for the interaction among the systems that form the network. Network architecture is the set of layers that defines both the protocols and physical systems that allow the communication among different systems. TCP/IP is the architecture that has been the base for Internet. This architecture has shown that it is simple and presents high protocol scalability[47] due to the nature of the functions defined in each layer. Despite of the good performance of this architecture in wired networks, it is not good for wireless networks nor for MANETs, where the environment conditions and the own features must be appropriately managed.

In a TCP path, according to the IEEE 802.11 standard, the MAC protocol reports a link failure to the routing protocol after a transmission and three retransmissions of a packet have failed, but it can be a false link failure, because in mobile ad hoc networks transmission failures may result from collisions, route failures, congestion and mobility[28, 40, 48, 110]. The TCP poor performance[48], can be improve using a cross layer approach. Some proposals, for example, suggest a direct communication between the physical layer and the network layer, where a true link failure is reported when a neighbor is out of range[48].

Next section explains the single layer architecture in MANETs and then is presented a brief exploration of cross layer architecture .

1.3.1 Single Layer Architecture

In MANETs, the layer's functions of the single layer architecture of the Internet model (TCP/IP model) are maintained, but adapted to the MANETs environment conditions. In this section, the functions of the architecture layers are specified using the OSI reference model.

Application Layer

The application layer is responsible to provide the user with network access, through the protocols and applications, to send and receive data[71, 85]. In MANETs there can be situations where certain nodes can provide access to specific services. In such cases and due to nodes mobility, services will not be always available. Thus the application layer must provide the user with information about the currently available services[71, 75].

Presentation Layer

The presentation layer is related with the data representation, syntax and semantics. The code formatting used in this layer depends on the defined structure that comes with the data, thus the data can be reorganized making it readable by the end application. Compression,

decompression, encryption and decryption, are processes accomplished in this layer, allowing the reduction of transmitted data and increasing the security, respectively[71, 85]. Presentation Layer is very important in MANETs to ensure security objectives, like privacy and authentication[49].

Session Layer

The session and data exchange control is the main function of the session layer. This layer offers services to start and finish conversations, and others like defining the next turn to transmit data[71]. In MANETs this layer is responsible for managing sessions, and has to take into account node mobility and resource allocation for applications.

Transport Layer

The transport layer is responsible for ensuring data integrity. To achieve this, the transport layer offers a group of services in charge of: ordering packets, multiplexing, segmenting, error detection, error recovery and flow control[71]. These services depend on the kind of transport protocol used, which can be connection or connectionless oriented. TCP is the transport layer protocol more reliable in wired networks and was created under assumptions of low bit error rates and low latency[90], but in MANETs the open medium increase the occurrence of retransmissions due to collisions and congestion, and the movement of nodes causes out of order delivery and large delays. That is the main reason for researches to propose and use an adapted TCP approach[56]. Some others state that TCP is definitely no well suited for MANETs and prefer to use UDP[90], but UDP is a non reliable connectionless protocol. Because of this, when data reliability is a requirement and UDP is used as the transport protocol, in the application layer a reliable transport is implemented[90].

Although the changes that have been made in some traditional protocols, MANETs are fully compatible with the UDP and TCP transport protocols, because the TCP/IP protocol stack is also used in MANETs[32].

Network Layer

Protocols in this layer, define how packets travel through the network from source to destination, depending on the cost. It might be a function of delay, number of hops, cost, or a mixture of them giving each one a weight according to the desired objective[1]. This layer uses addressing to identify host and routes[81], and to support routing assessments.

Routing in Mobile Ad Hoc Networks

MANETs features introduce complexity to the routing processes, because they imply that the network topology changes rapidly[57], so a route may be lost in few seconds.

In MANETs the routing has been a widely studied field, and the amount of proposed protocols is large. Routing protocols in MANETs can be classified using their behavior as a criterion[35]. The behavior makes reference to the criterion that are used by nodes to route a packet. Due to the large quantity of routing algorithms and the objective of this thesis, it is not a pretension to give a specific description of each routing protocol but to give elements to differentiate the strategy used by the most known protocols. Some researchers have proposed taxonomies[35, 51], but here instead of explaining taxonomies, the main criterion used to classify them are clearly stayed.

Type of cast

The type of cast used to forward a packet, is determined by the destination address[2].

- **Unicast routing protocols** The sent packets, are generated in one station and received by other station[2].
- **Multicast routing protocols** Packets are sent to a group of nodes, identified by a common address[2].
- **Geocast routing protocols** That protocols can be seen as a part of multicast protocols. The difference is that the destination group is located at a specific geographical area[51] compared with multicast protocols where the nodes of a multicast group can be distributed through the network. In geocast traffic is directed to a specific geographical area, i.e. GeoTORA[50].
- **Broadcast routing protocols** Delivered packets are directed to all nodes that composed the network[2].
- **Anycast routing protocols** A packet is intended to be delivered to one of the nearest groups of hosts. There is a group of hosts who share a common anycast address. When a package is sent to the anycast address, a route must be found to anyone of then[69].

Main works on routing protocols for MANETs establish a different taxonomy for unicast and multicast routing[51, 54]. However the same criterions to classify unicast protocols can be used to classify multicast protocols[54] and other type of cast protocols. In this work, a special emphasis is done when the classification criterion is only for multicast protocols.

Time to update or discover routes

Traditionally, routing protocols for MANETs has been classified based on the time when the protocols react to route invalidation[24, 32], thus protocols can be reactive, proactive, or hybrid:

Proactive routing protocols try to maintain a full knowledge of the entire network at all time[10, 86]. For this purpose, the routing protocol is always aware of the changes happening inside its neighborhood. In such case, the routing table is updated and the changes are reported to the entire network[65, 71]. Since mobility in MANETs is unpredictable and

frequent, there are large quantities of updates in the network. Therefore, when the MANET environment presents high mobility rate or the network is large enough [1], this kind of protocols are no well suited due to the amount of traffic flowing through the network[71, 33, 10] and the short amount of time that routes persist before being expired[65]. The main advantage of proactive protocols is that delays related to route discovery are low[65] since routing information is always updated. Proactive protocols are also known as table driven in the literature[10, 33]. Some proactive routing protocols are: DSDV, OLSR, OSPF - MANET, WRP, DSDV, FSR, AMRoute and AMRIS[24, 54].

Opposite to proactive routing protocols, *reactive routing protocols* only have information about the routes that are currently in used. So when a packet is to be send, the source request for a route to reach the destination, and the routing protocol proceeds to inquire the network by the route to use to deliver the packet[65, 86]. Thus, these types of protocols are also known as on-demand protocols or source initiated protocols because the routes are only discovered when they are asked by a source[10, 33, 71, 86]. The traffic routes can be discovered per-session or per-packet, being more mobility aware the per-packet basis, but implying that the amount of traffic will be higher[71]. Since reactive protocols do not need to be updating the routing information each time there is a change in the topology network, there will not be a constant traffic overhead[33] but the delay can be large[65]. AODV, DSR, ODMRP and MAODV are examples of reactive routing protocols[10, 24, 54].

One can no state which is better between proactive and reactive routing protocols, because what is an advantage in one of them is a disadvantage in the other one. For instance, proactive protocols do not suffer of high latencies, and have a global knowledge of the network topology, but the traffic overhead is high despite the routes are used or not. On the other hand, in reactive protocols the known paths are used and the traffic overhead is low, but exhibit significant latencies and the network topology is never known[57]. Consequently a type of protocol named *hybrid routing protocols* has been proposed. Hybrid protocols combine the best features of both proactive and reactive protocols. DST, ZRP, ZHLS, HARP and DST are some of the hybrid routing protocols mentioned in the literature[71, 54].

Network structure

If the network structure is taken into account to classify the routing protocols, then the function or role of each node must be inspected. Under this criterion, two classes of protocols exist: uniform and non-uniform routing protocols[54].

- **Uniform routing protocols**

  In that kind of protocols, nodes are homogeneous in their functionality on the routing processes[51]. WRP, DSR, AODV, DSDV can be classified in this category[54].

- **Non-uniform routing protocols**

[1] Small: 2-30 MANET peers, moderate: 30-100 MANET peers, large: 100-1000 MANET peers, very large: larger than 1000 MANET peers[23]

Protocols corresponding to this classification selects some nodes or group of nodes to realize special functions in the routing processes[54] in other to improve the routing. These protocols are normally used in large networks, due to the high collision rate and the scarce bandwidth these networks present. According to the characteristics of the groups formed, this class of protocols can be divided into zone based, cluster based and core based protocols.

Zone based hierarchical protocols, are those which organization is based on forming groups of nodes called zones[54] in order to limit the consequences of nodes movement to a reduce scope avoiding a global topology reorganization[35]. Usually, each zone has a gateway to realize the communication among nodes of different zones. The intra-zone communication is a local process that does not affect the entire network. ARP, ZHLS are zone based hierarchical protocols[54].

Other protocols, implement the routing by grouping nodes to improve behavior of the network[108], but also select a node to be the leader of the group and responsible for the management of the group members[54]. Such management functions include saving energy, control the membership to reduce the overhead in a group or to form groups with similar movement patterns, among others[108]. The cluster head is also in charge of the routing of all the packets in the cluster, implying a unique point of failure. These types of protocols are known as *cluster based hierarchical routing protocols* and examples of them are CGSR and HSR[54].

The *Core based multicast protocols* use structures that are composed of nodes that have special functions in the network. Some functions may be multicast data distribution and membership management. Here the CTB and AMRIS protocols can be mentioned[54].

It's important to have into account that in zone based and cluster based protocols node groups that are geographically close. Core based protocols select nodes that may be distributed all over the network but are critical to accomplish the complete network functions. Special functions are delegated to those nodes, and they compose the backbone of the network. CEDAR is a core based protocol[108].

Communication model

A routing protocol can use a single channel to transmit data packets and control packets, other are designed to transmit data packets in a different channel from the control packets channel. Thus, protocols can be classified as *single channel* or *multiple channel routing protocols*.

Network metrics

Routing protocols choose paths using a metric or construct a cost function using a mixture of metrics[57] depending on the objectives concerned with the network. The most used metrics and some of the protocols that use them are listed below:

- Hop number: number of hops or nodes that a packet must travel through before reach

the destination[54]. This metric is used in protocols such us AODV, DSR, OLSR among others[109].

- Link stability: the strength of the signals indicates how stable a link is. ABR, SSR and CEDAR use the link stability criteria[54].
- Link capacity: relative to link bandwidth and packet loss rate.[51, 54].CEDAR uses the link capacity as metric[54].
- Link utilization[51] and congestion. For example MOSPF protocol[54].
- Delay[54]: the main use of this metric is in the protocols that offers the enhancement of QoS for the traffic in the MANETs, such as AODV, DSR and OLSR[109].
- Energy: protocols that use this metric look for the minimization of the energy consumption. For example, the MPR protocol[51].

Network state information

The type of information saved in the routing tables, is also a criterion for classification[51]. According to this criterion, the following types of protocols are listed:

- **Topology based protocols:** Nodes route packets through the network based on the network topology information[54]. For instance, DSR that is a topology based routing algorithm, includes in the packet the complete route information from the source to the destination[54], thanks to the full network topology knowledge.
- **Destination based protocols:** these protocols, do not have complete network topology information, their routing tables have information about the next hop for sending packets to a specific destination. DSDV is a destination based protocol[54].
- **Location based protocols:** if this type of protocols is implemented in a network or part of this, the corresponding nodes require to be properly equipped to establish communication with some locating system as GPS. The position of the destination is used to define the route to forward packets[54, 71]. Some protocols establish the routing paths taking into account the movement velocity, but in order to accomplish that, they need to have access to a locating system. For this reason, they are included in the location based protocols class. LAR and DREAM are protocols in this category[54].

Content based multicast protocols

All the protocols cannot be used in any environment, thus the creation of a new routing protocol normally obey to the needs of an environment or its applications. Content based protocols route packets or choose the destination set based on the data content instead of a predefined criteria[54, 113]. CBM was proposed by Zhou and Singh[113] as a Content based routing protocol[113].

Tree based multicast protocols

Those protocols construct a multicast tree for each source node to achieve the packet distribution, thus multicast packets are forwarded using the tree until every multicast group node is reached and the packaged read[51]. This case is named *source rooted tree based multicast protocols*. *Core rooted tree multicast protocols* construct trees that are rooted in nodes with special functions. The trees connect all multicast groups of the network and their members[51]. AMRoute, CTB and AMRIS are examples of this kind of protocols[54].

Mesh based multicast protocols

Mesh structures are composed by nodes that can achieve connection through redundant paths, providing failure tolerance. That is why a mesh structure can be said to be more robust than a tree structure. Mesh based multicast protocols, use this type of structures to attain the distribution of the packets[51, 54]. Instances of this kind of protocols are ODMRP and CAMP[54].

Group forwarding based multicast

This class of protocols, choose a set of nodes to be responsible for the packet distribution for a multicast group, only those nodes can forward multicast packets[51, 54].ODMRP[54].

As can be seen, there are a lot of protocols that can be implemented in a MANET environment, but the big majority are not standardized. The MANET working group, have accepted as experimental Request For Comments (RFC) the protocols AODV (RFC 3561), OLSR (RFC 3626), TBRPF (RFC 3684) and DSR (RFC 4728)[59].

Addressing in Mobile Ad Hoc Networks

In MANETs, as in wired networks, each node needs to be identified with an address with purposes of communication and packet forwarding. The addressing scheme must guarantee that all nodes are available even if they move. Using the traditional wired addressing suppose that all the nodes have addresses that are members of the same subnet or network address space, so the destination nodes will not be property found because the assumption of the traditional routing strategies that says that the nodes with addresses of the same subnet are available on a single one hop network segment does not apply in MANETs[86]. Thus, the first approach for the MANETs addressing was to configure the nodes using independent address for each one, as a result each node performed host and router tasks.

Given the above statement, in the first stages of the MANETs development it was thought that addressing aggregation was impossible due to the movement of the nodes, but recent works, show how addressing aggregation may be accomplished while reducing overhead and delays, and doing a better use of node's memory with few routing table entries[80]. These approaches differ between them by the nodes that are involved in the routing decision processes. In[80] the development was done for AODV protocol, requiring the design of a special process for the conformation of subnets inside the MANET and implying changes in the nodes to support address aggregation. This process is not automatic, in addition the conformation of the subnet and the election of the subnet router are explained as a static process.

Currently, the configuration of the MANET nodes is an automatic process[18]. Some mechanisms ensure the uniqueness of the address for nodes that belong to connected MANETs (the ones that can communicate with other Internet nodes through a Internet Gateway) by using a pool of addresses that are managed by a gateway Internet node, which receives DHCP request from the client nodes[5]. Others authors propose to use a special address format for the identification of each node into MANET. Jelger[42] suggest to use the Unique Local IPv6 Unicast Addresses format that was proposed by Hinden and Haberman in the RFC 4193[39], with this format each node generates its own address guarantying the uniqueness due to the use of a random number of 56 bits, so Jelgel affirms that these addresses have a very low collision probability.

With regard to the multicast address spaces used for MANETs the RFC 5498 have established that all the nodes that are in charge of the routing process, will be attached to the multicast addresses 224.0.0.109 for IPV4 and FF02:0:0:0:0:0:0:6D for IPV6. Those multicast addresses receive the name LL-MANET-Routers[22].

Data Link Layer

In the wired networks the Data Link Layer is recognized to be the layer responsible of the physical addressing by the use of the MAC addresses. This layer has the ability to detect and notify errors, control the flow and avoid the collision excess by the management of the access to the media, known like Media Access Control[85].

In MANETs all those tasks are the same than in wired networks, but due to the challenges imposed by movement of the nodes, MANET researches have done special emphasis on some of them.

Opposite to wired networks, MANETs have not a static topology[72]. The topology of a network is the set of links that can be used for the data transmission[112]. MANET devices interfaces characteristics imply that the topology is not predefined because the signals travel open media and if any device is inside the network coverage area it can listen and participate in the communication. On the other hand mobility and noise create and destroy connections continuously, and, finally, links are not always symmetric. This behavior and the absence of a fixed infrastructure are the reasons that MANETs are known as dynamic topology networks.

Routing protocols use network topology to make routing decisions for packets in MANETs. Since the topology is dynamic, it is necessary that topology control protocols take the initial network and generate and maintain a connected network that match, at least approximated, certain conditions, like connectivity, energy-efficiency, throughput, and robustness to mobility[72] but taking into account the nodes mobility, so, the communication can be optimized in order to improve the functions carried out by the upper protocols that are implemented, in special the Routing Protocols[76]. In this thesis, Topology Control Protocols have been located in the Data Link Layer of the OSI model, however this discussion continues opened due to the exchange of information between the Data Link Layer Protocols and the Network Layer Protocols with the Topology Control Protocols[76].

Basically the Topology Control Protocols take the physical topology of a network with directed and undirected links, and composes a logical topology by the management of the transmission powers. The resulting links will be used for node communication[76, 112]. Thus what is achieved is the hiding of the multiple physical hops from the Network Layer protocols, but without possibilities of avoiding the noise and the interference produced in the physical topology among other performance problems[23].

In addition to the problems related to the network topology, there are some problems related to the technologies in MANETs. As is mentioned earlier due to the fact that any wireless device with MANET capabilities can be part of a MANET, there can be a mixture of technologies, but in order to accomplish communication, the interfaces of the devices that belong to a MANET need to have the same addressing scheme at the MAC layer and also, to make things easier, the frame sizes of all the technologies used should not vary[23].

Another important topic concerning Data Link Layer is the way that nodes access the medium. In networks where the devices interfaces have the ability to listen while transmitting the mechanism to access is the sense of the medium with collision detection or CSMA/CD, but in MANETs the devices could not do that (at least until recent years[78]), so in this kind of networks the medium is sensed with collision avoidance or CSMA/CA[110]. The CSMA/CA mechanism, first developed for Infrastructure Based Wireless Networks, consisted in the exchange of a couple of control packets; when a node has data to transmit it first listens the medium and if it does not detect any transmission, then sends a Request to Send packet (RTS) to the destination, when the destination receives the RTS sends a CTS indicating the source that can start to send the data packets[45, 71]. The CSMA/CA mechanism is useful for avoiding collisions, due to the fact that the nodes that listen a RTS or a CTS packet are inhibited to transmit for a specific amount of time, but it leaves two new problems: hidden and exposed terminal problems[41]. So, some changes was done over the control packets and MACA and MACAW protocols were introduced to solve the hidden and exposed terminal problems, which also implied that the new mechanism consisted in the exchange of the RTS and CTS with the amount of data to be send, and ACK packets of each successful sent data packet[45]. Today, the 802.11 standard has improved the performance of the medium access control for MANETs (it has become the most used standard in MANETs deployments[32]) with new protocols that permit the use of multiple channels over the same medium[71, 110] and to mark data packets with a priority to offer Quality of Service in this networks[52]. As was mentioned above, medium access mechanisms have evolved to make possible the collision detection in wireless networks[70, 78].

Physical Layer

The physical layer manages the physical, electrical and mechanical characteristics of the network and its nodes. For example the channel coding, radio transmission, modulation, frequency range, carrier sensing, among others[32, 71]. A MANET can be formed with any kind of devices, or a mixture of devices with different computational and communicational

capacities, in fact the collection of host need not to be homogeneous[71], but is a must that all the devices work with the same technology (which can be Bluetooth, IEEE 802.15.3 (WPAN), HiperLAN1, HiperLAN2, IEEE 802.11 and HomeRF, among others[71]) to achieve communication. Due that the physical specifications depends on the technology used, we will not state all their characteristics, but they can be review using the different standards and books like[71].

1.3.2 Cross Layer Architecture

As was seen in the previous subsection, the behavior and characteristics of MANETs are different from those one wired networks, and some protocols have been changed to achieve a good adaptation to those new features. Indeed the architecture has needed to be reconstructed. Cross Layer Architecture, is the proposed architecture for MANETs, and it involves a framework that permits the exchange of useful information among the traditional layers that were mentioned above. To clarify the need for Cross Layer Architecture is enough to mention some examples of its utility. For instance, a packet dropper would be more easily detected if the Session Layer could know how many packets have been dropped[31], or a node can make better assumptions about the reachability of a node when a packet could not been sent successfully because sometimes it could be produced by a failure in the packet delivering[23]. In this sense, some authors affirm that the functions and processes in MANETs can be improved by the use of an architecture where layers do not execute their tasks in isolation[47, 3].

Cross Layer Architecture has been developed in order to have an efficient exchange of information among the traditional layers. The first step is to identify the interdependencies between layers and the information that must be shared[23, 47], and then mirroring these relations in a non complex architecture in order to conserve the compatibility with the current systems and networks[47]. The Cross Layer Design gives a more efficient communication among layers, but this job must be carefully done, due to the fact that loops can be formed, what would imply instability in the system[47].

Some Cross Layer Architectures have been developed for special types of traffic or applications[15, 79], but literature suggest that a generic architecture is needed in order to give support to all applications[47].

1.4 Main Challenges and Problems in MANETs

Pretending to give a clear idea of the necessity to use an Intrusion Detection System in MANETs, the security vulnerabilities are presented, in this section.

The main problems that are present in MANETs, results from the distributed structure of this kind of networks. The distributed structure and the open medium mean that any new device can enter to the network and listen the communication that is taking place, and do

not permit to have clear physical perimeter or line of defense, as in wired networks, where the administrator knows where to locate the entities to secure the network[8, 11].

For the sake of their good operation, MANET protocols require the cooperation and confidence among the nodes of the network[102]. One of the clearest examples resides in the routing protocols, where nodes ask their neighbors for routes to a specific address, but if an intruder is into the network, it can advise himself as the destination or give a false route. That is why some authors argue, like [102], that the cooperation is needed but is also a weakness that can be used by the intruders to cause network misbehaviour.

Finally, in some MANETs, devices must be light weight and not too big, to grant that a person can carry it easily. But the small size is also a disadvantage, because is difficult to provide physical security to such as device. So the lack of physical security makes the nodes prone to be stolen, and therefore the network security would be in risk.

Next chapter presents Intrusion Detection Systems and their architecture.

Chapter 2

Architecture of Intrusion Detection Systems in Mobile Ad Hoc Networks

Security systems can be classified as proactive and reactive. The proactive systems are those that establish confidence relationships and try to protect the system against the intrusions, for example, cryptography and authentication [82]. The reactive systems are those used to react when an intrusion makes presence in the network, for example Intrusion Detection Systems. In this chapter, an exploration on the IDS architectures for MANETs is made.

The characteristics of MANETs impose severe restrictions that do not permit the use of traditional IDS for wired networks over MANETs. First, due to the lack of a clear line of defence the attacks can come from anywhere in the network, and second the distributed infrastructure removes the existence of a certain points of the network where the traffic can be centralized, filtered and analyzed [111].

Flat MANETs were the first studied networks for IDS implementation, then the limitation of resources and the kind of attacks showed that nodes are like members of a community, where the hierarchy and collaborative work improves the efficiency in the intrusion detection. This section presents a short explanation of the architectures that have been proposed for MANETs.

IDS architectures for MANETs can be divided in the following categories [8, 74]:

- Stand-alone
- Distributed and Cooperative
- Hierarchical

2.1 Stand-alone Architecture

In this architecture each node is responsible by its own security, and the state of the neighbors is unknown. Each node runs an IDS, and if a intrusion is detected then itself must make a

decision about how to act against it [8, 74]. The use of an IDS in each node increase the consumption of node resources, and although the security is spread in every node this scheme does not permit to infer about the state of the entire network due to the fact that each event is analyzed independently.

2.2 Distributed and Cooperative Architecture

The responsibility of intrusion detection is shared with the neighbors when necessary. In this architecture, each node runs an IDS or an IDS agent (depending on the technology used) that can detect intrusions and make its own decisions. Only when the evidence of intrusion is inconclusive the neighbors start a global intrusion detection [8, 74].

2.3 Hierarchical Architecture

This works like an extension of the Distributed and Cooperative architecture because, although the network is divided into clusters (groups of nodes), cooperation among nodes is present. Each cluster member is responsible for the monitoring and decisions concerning to the system and user activity, and the cluster head must also monitor the packets that crosses its cluster [8, 74].

Most of the developed IDS strategies and proposals in the literature are based on the Distributed and Cooperative, and Hierarchical architecture. Some use a complete IDS on each node, and others divide the IDS tasks and assign functions through IDS agents. Agents present a higher fault tolerance for such a frequently changing environment.

The proposal of Zhang and Lee [111] for example uses a Distributed and Cooperative approach. The cooperation among nodes takes place when an intrusion is detected locally by a node, or when the evidence is inconclusive and requires a deeper analysis. On the other hand, Rezaul et Al., in [74], continue with the proposal of Zhang and Lee, but impose some changes in the cooperation among nodes and the technique to detect the intrusion. When a node detects a possible intrusion it sends an alert to its neighbors to initiate the cooperation procedure, and each one calculate the probability of the malicious action. Then all probabilities are combined and finally a response is initiated. The technique used to combine the probabilities in order to classify and discover the intrusions is the Naïve Bayes approach.

An approach that mixes the Distributed and Cooperative with the Hierarchical architecture can be found in the work of Song et Al., in [82]. Although they do not create little groups of nodes or clusters to be classified as a pure hierarchical architecture, they use a Connected Dominating Set (see the definition in chapter 9) to locate the network monitors. The established difference among the nodes imply a hierarchical approach. In this scheme, each node executes a host based monitor for the system and user applications, and the net-

work monitors are executed only in the Connected Dominating Set nodes, asseverating that the connected set of monitors allows that monitors can analyze each other.

Nargunam and Sebastian, in [67], propose the formation of clusters without cluster heads to avoid the existence of a point of failure. In the scheme all cluster members monitor each other. The dynamic of the network is supported by the proposal. This scheme can be seen like a cooperative approach but the division into groups allows the classification as a hierarchical architecture too.

Kachirski and Guha, in [44], developed a scheme that defines a hierarchy by the use of clusters and cluster heads. The IDS is divided into its functions, which are then assigned to nodes by the use of agents. Every node in the network, has a monitoring agent that, depending on the hierarchical position of the node, can work as a host monitor or a network monitor. The cluster heads are the nodes where the monitoring agents run as both host and network monitor, and are the responsible of taking decisions based on the analysis of the packets that are originated by the nodes of the cluster. A similar strategy is proposed by Sterne and Balasubramanyam in [84], but the cluster heads are differenced among them because they are classified by levels. In each level the cluster head summarize and correlate the events of the lower level cluster heads until find something conclusive to make a decision and start a response.

Chapter 3

Distributed Algorithms for Leader Election

Distributed Algorithms for Leader Election are those algorithms implemented over networks with the objective to elect a global leader or local leader depending on the purpose, and obey to the condition that all the nodes of the network are involved in the election.

When talking about that kind of algorithms, one of the first references that are consulted is Nancy Lynch [58], whom in her book Distributed Algorithms discusses the techniques used by some distributed algorithms. Those techniques can be used as a guide in the development of distributed algorithms. For leader election, some of the techniques (based on the nodes identification) that are mentioned work for networks under certain topology conditions or topology knowledge. For example LCR, HS and AsyncLCS are designed for rings; Flood and OptFlood need to know the diameter of the graph. These topology conditions and the imposed by the other techniques mentioned in the book are very severe, especially when the topology changes frequently, thus is impracticable to apply them in MANETs for the leader election.

In [61] two algorithms are presented, both of them for synchronous networks and with some limitations in the nodes movement. The leader is elected depending on the height that is related to the identification of the nodes.

Other strategies used in the election of local leaders, like for example in clustering. In addition to the node ID, they also have into account variables as node degree, mobility behavior, energy consumption, battery energy, etc., or a mixture among them. Some clustering schemes used in MANETs are analyzed in [108].

Lots of algorithms have been proposed for Leader Election, but basically the strategies are the mentioned above.

Chapter 4

Distributed Algorithms to Find the Central Point in a Network

The problem of finding the Central Point of a Network is the reduction of the Minimum Diameter Spanning Tree (MDST) problem[21] (see the definition in chapter 5). This section presents the strategy used by some algorithms to find the central point in order to construct a MDST.

The technique to find the central point of a graph, consists basically in finding the shortest path from a vertex (say *vertex A*) to every other vertex that composes the graph. Once that the distance from vertex A to every other vertex has been calculated, the longest of them is chosen as the eccentricity of vertex A. This procedure must be done for each vertex in the graph. When all the vertices have their eccentricity, the vertex with the smallest eccentricity is chosen as the central point of the graph. Using Floyd's algorithm the shortest paths can be calculated, then the eccentricity of each node is calculated and the central point of the graph is selected as the one with smallest eccentricity. In [106], Yang et al. compress the graph and then apply the Floyd's algorithm, so they can achieve better complexities than using directly the Floyd's algorithm. Although their proposed algorithm is said to be designed to networks, it cannot be implemented in MANETs because the strategy is centralized and the network assumed is static. The goal in [106] is to find the diameter of the network, but is related to this chapter because it is equivalent to find the central point.

A distributed algorithm is presented in [21]. The authors established that each node must calculate its distance to every other node in the network by sending messages to each neighbor and waiting until all its active neighbors send a message updating the distances. The algorithm is designed for a static network, but the approach used to find the distance of a node allows the addition of a new node to the network to be managed without problems, but it implies some changes in the quantity of messages that are originated.

Chapter 5

Graph Theory Definitions

Graph theory provides the theoretical background and the common language for network analysis. In this chapter we present the definitions that are relevant for the work.

5.1 General Definitions

Graph $G(V, E)$**:** is a collection of a finite set of vertices V and a finite set of edges $E = \{(v_i, v_j)/v_i, v_j \in V\}$, where vertices are represented by points or nodes, and the edges are represented by lines that connect points [95, 21].

Distance $d(u, v) = d_G(u, v)$**:** is the minimum length (weight) of the paths that connect two nodes in a graph [16, 21].

Eccentricity $ecc(v)$**:** is the largest distance from a vertex v to any other vertex in V, i.e., $ecc(v) = max_{u \in V} d(u, v)$ [98, 21].

Diameter $D = D(G)$**:** is highest eccentricity of any graph vertex, in other words is the longest distance between any two graph vertices of a graph, i.e., $D = max_{v \in V} ecc(v)$. It is also known as the "longest shortest path" [97].

Radius $R = R(G)$**:** is the minimum eccentricity of any graph vertex, i.e., $R = min_{v \in V} ecc(v)$.

Central point or absolute centre: is the node whose eccentricity equals the graph radius, i.e., v is the central point of G if $ecc(v) = R(G)$. In some cases, there is not a unique central point, so the set of all central points is called the graph centre [92, 96, 21].

5.2 Spanning Trees

Let $G(V, E)$ be a graph, then:

Spanning Tree (ST): a spanning tree of G is a subset E' of $n-1$ edges from E that form a tree connecting all vertices of V [101].

Minimum Spanning Tree (MST): a minimum spanning tree of G is a ST whose total sum of edge weights is the smallest possible. The STs presents a special case when the graph is unweighted, due to the fact that any spanning tree is a minimum spanning tree [100]. A MST can be calculated using Prim's or Kruskal's algorithm. The procedure of Prim's algorithm basically consists on the selection of any vertex for the tree, and then at each iteration the vertex with the lowest weight edge that connects the tree with vertices that are outside is added to the tree. The Kruskal's algorithm procedure is similar, but in this case, the edge with the lower weight is selected and at each iteration the next lower edge is added to the tree taking into account that the tree cannot have loops.

Shortest Paths Tree (SPT): is a spanning tree constructed to connect a vertex to every other vertex in the graph, with the condition that the sum of the edge lengths from the initial vertex to each node is minimized [34, 107]. The strategy used by the algorithms like Dijkstra is to start with a particular vertex and find the shortest path from it to every other vertex in the graph.

Minimum Diameter Spanning Tree (MDST): is a ST of minimum diameter. The main characteristics of this tree is that the root of the tree is the vertex considered as the absolute centre of the graph [37] and the paths from the root to every vertex of the graph are the shortest paths. Thus, the SPT whose root is the absolute centre is a MDST of G [21].

Chapter 6

Petri Nets

Petri Nets is a tool used to analyse and model system processes and its dynamics, both graphically and mathematically. The evolution of the first kind of Petri Nets, has been the trigger to have as many systems where they can be applied, some examples of these systems are: distributed-software systems [6], distributed-database systems, concurrent and parallel programs, flexible manufacturing/industrial control systems, discrete-event systems, etc [66].

A Petri Net is formally defined as a tuple, such that $PN = (P, T, F, W, M_0)$, where [36, 66]:

$P = \{p_1, p_2, .., p_m\}$ is a finite set of places, the representation of each place is a circle,
$T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions, the representation of each transition is a bar,
$F : (P \times T) \cup (T \times P)$ is the set of arcs, where $(P \times T)$ represents the arcs directed from the places to the transitions, and $(T \times P)$ represents the arcs directed from the transitions to the places,
$W : F \rightarrow \{1, 2, 3, ...\}$ is a weight function ,
$M_0 : P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking,
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

The following notation is important in order to understand some definitions that are given bellow.

${}^\bullet t$ are the set of input places of a transition t,
$t^\bullet$ are the set of output places of a transition t,
${}^\bullet p$ are the set of input transitions of a place p,
$t^\bullet$ are the set of output transitions of a place p.

Petri Nets are composed by events and conditions. The transitions represent the events, which are detected in the systems by sensors or also because some variable that is being

monitored has changed. The places, also known as preconditions and consequences [36], can be seen as the outputs of the system, these outputs are the actions executed by the actuators of a system, and in computer science may be an action over a variable, the sending or dropping of a message, etc. Figure 6.1 is a Petri net of a simple filling system, the p_0 place represents the opening of a water valve, and the transition t_0 represents the desired level of filling.

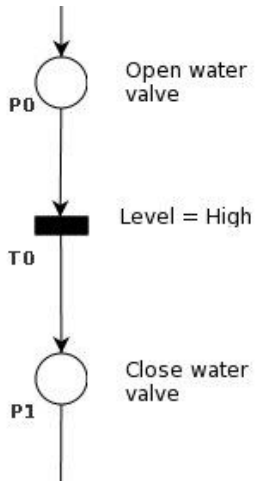

Figure 6.1: Petri net of a filling system

Petri Nets are also composed of tokens. The tokens are located in the places or conditions, and the presence or absence of tokens in the places of a net in a given time is called the marking. The marking represent the current state of the system. For example, in the net of Figure 6.2, if a token is in *init* place the system is ready to start its process. In Figure 6.1, if there is a token in the p_0 place, then the valve will remain open until the sensor indicates that the desired level has been reached.

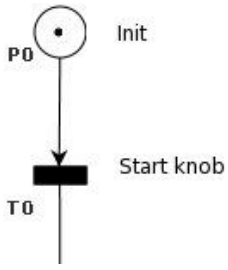

Figure 6.2: Petri net in its init place

A place can contain zero, one or more tokens, and they can be equal or different depending of what they are representing. Some systems do not need to make difference among its tokens,

the kind of Petri nets that represent these systems are called *low level* Petri nets. Low level Petri nets are the nets used in this thesis.

The tokens flow through the Petri net, pointing out the evolution of the real system. When all the preconditions related to an event are met, the system waits the event occurrence; this situation corresponds to the existence of the number of tokens indicated by the weight of each arc (if the arc does not have any number associated then it is assumed that the weight is 1) in the input places of a transition, so it is said that the transition is enabled. If an event occurs and the transition is enabled, the transition is fired and the number of tokens indicated by the weight of the arc directed from the place to the transition are removed from the input places, and the number of tokens indicated by the weight of the arc directed from the transition to the each output place are added [36]. See Figure 6.3.

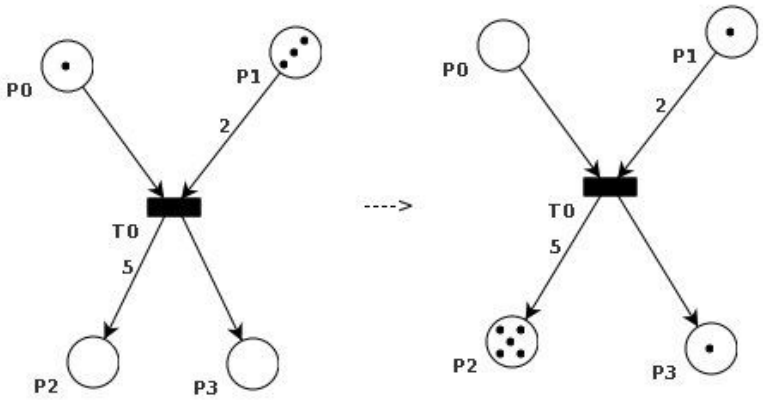

Figure 6.3: Firing of a transition

6.1 Petri nets classification and properties

According to the structural topology of the net and its dynamics, a Petri net can be classified in the following manner:

Ordinary: if all the arc weights of the Petri net are 1's, then the net is said to be ordinary [66], i.e. $\forall f \in F, W(f) \rightarrow \{0, 1\}$.

Pure: a Petri net is said to be pure if it has no self-loops, i.e., a place is not the input and output place for a transition [66], i.e., $\forall t \in T, {}^\bullet t \cap t^\bullet = \emptyset$.

Bounded: a Petri net is bounded if the number of tokens in each place is finite for any marking reachable, indicating that the system will not suffer overflow of data or resources, i.e., $\forall p \in P \wedge M \in R(M_0), M(p) \leq k$, where k is a finite number and $R(M_0)$ indicates that M is reachable from M_0. A subclass of the bounded nets are the *safe* nets whose places have 0 or 1 token for any marking [66]. Safe nets are also known as binary nets [83]. A Petri net can be also structurally bounded, making reference to the topological properties of the net, if the net is bounded for any finite initial marking.

Deadlock-free: a Petri net is said to be Deadlock-free if every reachable marking enables some transition [29].

Live: a Petri net is live if all its transitions are live [38], i.e., if for every reachable marking and every transition t it is possible to reach a marking that enables t [29].

Well-Formed: a Petri net is said to be well formed if it is bounded and live [29].

Reversible: these nets are identified because the initial marking or initial state can be reachable from any other marking of the net [66].

Conservative: a Petri net is said to be conservative if the sum of the number of tokens in the net keeps constant for any reachable marking [68].

Controllable: a Petri net is said to be completely controllable if any marking is reachable from any other marking [66].

6.2 Analysis methods

The analysis of Petri nets can be accomplished using the following techniques or methods, one of them allows the analysis of specific properties related to the initial marking, while the other allows having a vision of the structural properties of the net, and thus a system analysis.

6.2.1 Coverability Tree

This method is a state space analysis which consists in finding all possible markings or states that the net can reach from the initial marking. A coverability tree is a graph representation, where the nodes are the reached markings and the arcs are the transitions that are or have been enabled. The symbol w is used to represent a node representing a marking not bounded.

The coverability tree allows the study of behavioural properties like reachability, boundedness, safeness, liveness and conservation. The net is said to be bounded if the markings are bounded, i.e., if the symbol w does not appear in the nodes of the tree, and safe if the

markings are 0 or 1 for all the places, but if for a place all the markings are in 0 then it indicates reachability problems for the state that the place represents [36].

In addition to these properties, the coverability tree helps to know which transitions cannot be fired. If some transition does not appear in the arcs of the tree it means that the transition is not enabled during the net execution.

6.2.2 State Equation and the Incidence Matrix

This technique is an structural analysis, where the incidence matrix represents the relations among the places and the transitions in a Petri net, and the state equation use the incidence matrix to make inferences about the structural properties of the net (controllability, structural boundedness, structural liveness, repetitiveness, etc.). This method can be seen as the generalization of the coverability tree technique, due to the fact that the coverability tree is realized starting from an initial marking, but this structural analysis does not depend on an the initial marking and the results are valid regardless of the initial marking.

A more detailed explanation of these techniques can be seen in [66].

There are other techniques to analyse Petri nets, like the reduction and the simulation [66, 83], we also use it in the analysis of the Petri net for the algorithm, but is not necessary to make a deep explanation of them.

6.3 Special structures

6.3.1 Concurrency

This structure models events that can take place at the same time, or in any order without interfering, in other words, concurrent transitions are those which are enabled for a marking M but do not share an input condition or place [62].

6.3.2 Synchronization

This structure models a task that must wait to be executed until previous processes (at least two) have been completed. This is modelled by an event or transition that has more than one precondition.

6.3.3 Conflict

Two events are said to be in *conflict* when they have a common precondition and only one of them can occur but not both. It is modelled with two or more transitions that share at

least one precondition or place [36].

There are two types of conflicts: *symmetric* and *asymmetric conflicts*. To understand the difference between these conflicts is required to know a concept called *Enabling Degree*. The Enabling Degree of a transition is a quantitative measure, that informs how many consecutive times a transition can be enabled for a given marking, for example, for the net shown in Figure 6.4a) the Enabling Degree (ED) of transition t_0 is 2, but in Figure 6.4c) the ED of t_0 is 2 and has a value of 1 for t_1.

Going back to the types of conflicts that can be modelled with Petri nets, *symmetric conflicts* are those in which if two transitions are in conflict, said t_1 and t_2, firing t_1 will decrease the ED of t_2 and viceversa; contrary to *asymmetric conflicts* in which firing t_1 will decrease the ED of t_2, but the ED of t_1 will not be decreased by the firing of t_2 [14, 62]. Sometimes, the presence of these conflicts is due to the marking of the net or the structure of it [62]. See Figure 6.4.

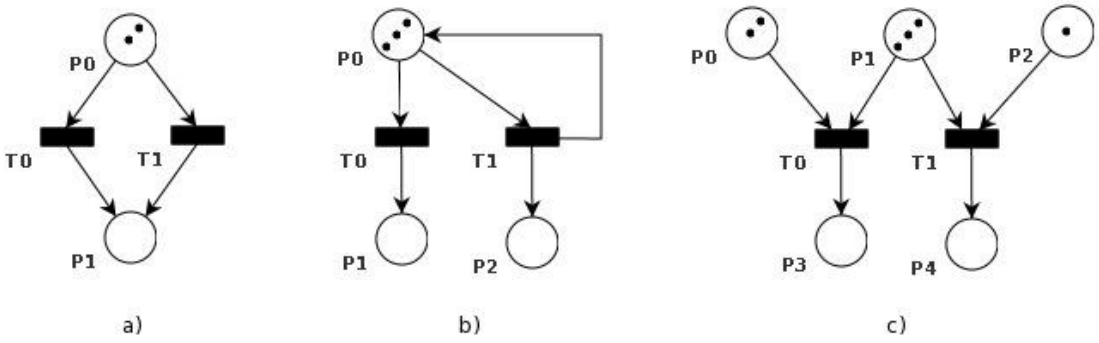

Figure 6.4: Symmetric and Asymmetric conflicts in a Petri net. a) Symmetric conflict, b) Asymmetric conflict, c) Marking dependent conflict.

In the net of Figure 6.4a) the ED of t_0 and t_1 is 2, if t_0 is fired the ED of t_1 is decreased to 1, and the same happens to t_0 if t_1 is fired. In Figure 6.4b) the ED for both t_0 and t_1 is 3, if t_0 fires the ED of t_1 will be decreased, but the firing of t_1 will not decrease the ED of t_0. In figure 6.4c) the type of conflict depends on the marking, for example, for the marking showed the value of the ED is 2 for t_0 and 1 for t_1, it does not matter which transition is fired because that event will not decrease the other transition's ED so the net have not conflict for this marking, but suppose that p_0 and p_2 have two tokens and p_0 has one token, so the firing of t_0 will decrease the ED of t_1 and viceversa, but in the case that the number of tokens is 3 for p_0, 2 for p_1 and 1 for p_2, the firing of t_0 will not decrease the ED of t_1 but the ED of t_0 will be decreased in the case that t_1 fires.

6.3.4 Confusion

Confusion is present in a net when concurrency and conflict are mixed [62]. In this case, there are also *symmetric* and *asymmetric confusion*. *Symmetric confusion* is in a Petri net when two transitions are concurrent (both can be enabled for a marking at the same time but they do not interfere) but no matter which of them is fired, there will be always a conflict with a third transition, see Figure 6.5. *Asymmetric confusion* is the structure that has two concurrent transitions, but the presence of a conflict will depends on their firing order; see Figure 6.6 [26, 73].

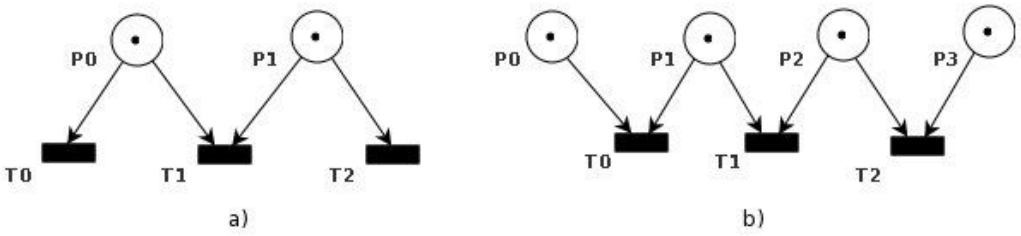

Figure 6.5: Symmetric confusion in a Petri net.

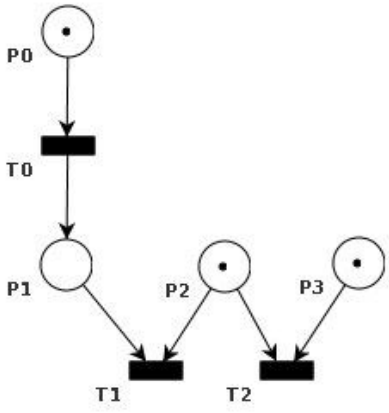

Figure 6.6: Asymmetric confusion in a Petri net.

Observe that Figure 6.4c) can be taken as a *confusion* but if it is careful analysed is easy to see that there is not a concurrency between t_0 and t_1 due that they interfere with each other.

6.4 Subclasses of Petri nets

Marked Graphs (MG): are ordinary Petri nets, whose places have exactly one input transition and exactly one output transition [66]. Marked graphs, allow the representation of synchronization of activities, but not of conflicts or decisions, which is the reason to be known also as Synchronization Graphs (SG) [38], thus a Petri net is identified as an MG if $\forall p \in P : |{}^\bullet p| = |p^\bullet| \leq 1$.

State Machines (SM): are ordinary Petri nets, whose transitions have exactly one input place and exactly one output place. State machines allow the representation of decisions, but not the synchronization of activities [66]. If $\forall t \in T : |{}^\bullet t| = |t^\bullet| \leq 1$, then the net is called a SM.

Free Choice (FC): are those nets where the all the output transitions of a place are enabled at the same time, so the choice is free for each place in the net. The FC admits the existence of symmetric conflict and synchronization, but does not allow to mix them [29]. A petri net is said to be a FC if $\forall p, q \in P, p \neq q \wedge p^\bullet \cap q^\bullet \neq \emptyset \Rightarrow |p^\bullet| = |q^\bullet| = 1$. Figure 6.7 shows two examples of the structures allowed into the FC Petri nets class.

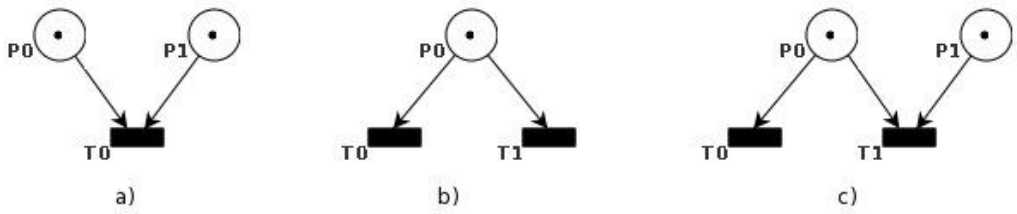

Figure 6.7: Free Choice Petri nets. Nets a) and b) are FC, but the net c) is not a FC Petri net.

Extended free choice (EFC): this class of Petri nets are as its name said an extension of the FC nets, the extension consist in the sharing of more than one output transition [29] with the condition that the set of input places must be the same for the transitions involved in the conflict. This allows mixing symmetric conflicts with synchronization. To clarify the concept there is the formal definition: a Petri net is said to be a EFC if $\forall p, q \in P, p^\bullet \cap q^\bullet \neq \emptyset \Rightarrow p^\bullet = q^\bullet$. See Figure 6.8.

Simple (S): a Simple Petri net has the characteristic that for each transition only one of its input places can be shared with other transition, i.e., each transition is involved in maximum one conflict at the same time [88]. Thus, a Petri net is simple if $\forall p, q \in P, p \neq q \wedge p^\bullet \cap q^\bullet \neq \emptyset \Rightarrow |p^\bullet| = 1 \vee |q^\bullet| = 1$. A graphical example is presented in Figure 6.9.

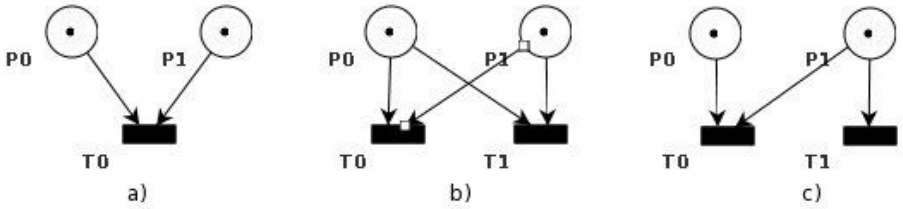

Figure 6.8: Extended Free Choice Petri nets. Nets a) and b) are EFC, but the net c) is not a EFC Petri net.

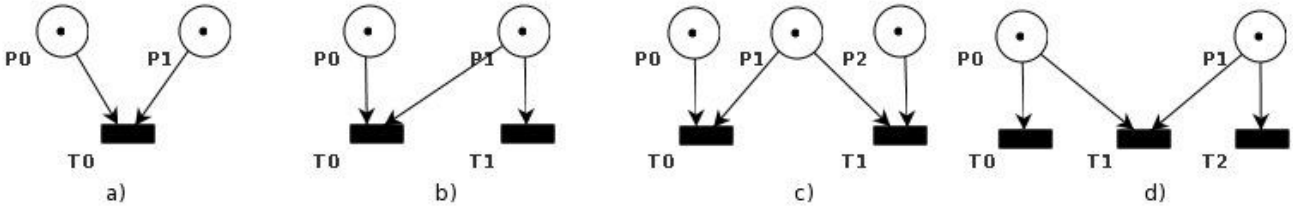

Figure 6.9: Simple Petri nets. Nets a), b) and c) are Simple, but the net d) does not meet the aforementioned characteristics.

Extended Simple (ES): in this class of Petri nets, each transition can be involved in maximum one conflict at the same time, but the input places to the transitions that are in conflict can have output sets of transitions with more than one element [38], i.e. a Petri net is said to be an ES if $\forall p, q \in P, p^\bullet \cap q^\bullet \neq \emptyset \Rightarrow p^\bullet \subseteq q^\bullet \vee q^\bullet \subseteq p^\bullet$. See Figure 6.10.

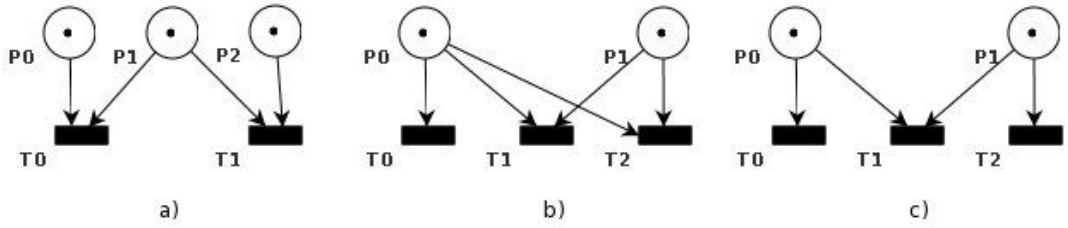

Figure 6.10: Extended Simple Petri nets. Nets a) and b) Extended Simple, but not the net d).

Chapter 7

Algorithm for the Leader Node Selection

In this chapter, the proposed algorithm is presented. There are some basic assumptions on the MANET that must be fulfilled for the algorithm to work.

7.1 Assumptions

The medium and the heterogeneous devices, make the relationships established with neighbors in a MANET to be dynamic and more complex than in wired networks [23]. One consequence of this dynamic is that sometimes the links are not bidirectional but unidirectional, so that communication between two nodes can be sometimes in one direction. For our purposes, bidirectional links are assumed.

Furthermore, it is assumed that all nodes have the same transmission range (thus the network can be seen as a unit disk graph [7, 55]), the hop to hop time is the same for all nodes, and all wireless links have the same probability of failure [54]. In addition, in order to improve the stability of the network and the *leader*, it is assumed that no nodes can be on or off; they can only enter or leave the network, ensuring that the members will not suddenly disappear.

The identification of each node is very important. For this reason, it is established that each node has a unique identifier (see section 1.3.1) that may be its IP address, MAC address or part of them. This identifier is known to the algorithm as the node ID.

In the development of this algorithm, it is assumed that a *Topology Control Protocol* is already implemented in the network, so the network topology is defined. The *Topology Control Protocol* is responsible for stating the physical links used for communication, defining the logical neighbors and preventing loops that may be present at the physical layer due to the medium conditions.

The security of the algorithm will be discussed later in another chapter, so for now honest exchange of messages between nodes is also assumed.

7.2 Proposed Strategy

The sensors are distributed on the nodes of the network, and are responsible for collecting and analyzing local events in their neighbourhood to infer the state of local security. The information is then sent to a network node which is responsible for correlating and analyzing data to find any signal of intrusion.

Although a single node to correlate events across the network implies the existence of a point of failure, the assumptions ensure that this node will not suddenly disappear, but it is necessary to find a location that enhances the stability of IDS and hence the network stability. For the rest of this thesis, this node will be known as the *leader* node.

Due to the fact that each sensor node must send the information to the *leader*, that the failure probability is the same for all links and the time spend by a packet to go from a node to its neighbour is the same for every pair of nodes in the network, the best position of the Leader is over the region of the network that suggests the minimization of the number of hops that the packets must travel through until find the *leader*, since the minimum path hop count for the routing will derive into a more stable network with lower quantity of collisions and traffic overhead [54].

The minimization of the number of hops for the entire network with regard to a node, can be extrapolated to find the minimization of the sum of distances of a reference point from the other points that are present in a defined space. According to Schlegel [77] the result of this minimization, indicates that the minimum is reached when the reference point coincides with the centroid (see page 25) of the points.

A network can be viewed as a graph where points are nodes and lines are links [95]. If this graph is taken, and each node's eccentricity is calculated using the shortest route, it would be easy to find the node with the smallest eccentricity. Using the found node as root, a tree from the original graph can be built. In this new tree the eccentricity can be used as the weight of each node, so the root coincides with the centroid of the built tree [94]. But the root of the tree is also the centre of the graph, due to the fact that its eccentricity equals the radius of the graph [93].

Now, the strategy is clear:

1. Calculate the eccentricity for every node in the network
2. Find the node with the lower eccentricity, and call it the *leader* or central node
3. Construct a tree with the *leader* as the root using the shortest paths to every node from the root.

At the end of the algorithm, the network will have only a unique *leader* for each connected set.

7.3 Design and definition of the Algorithm

The development and design of this algorithm was accomplished using the idea of "the simplest first", in which a simple case is taken into consideration and then it is solved, in the first place with a static topology and then with a dynamic topology. In this section the algorithm will be described in order to give a clear idea of how it works, so the reader can make good deductions on the algorithm behavior for some special cases.

7.3.1 A general description of the eccentricity calculation

The general behavior of the process of calculating the eccentricity is shown through the following example. Suppose the network of Figure 7.1.

Figure 7.1: Net with two nodes.

First, each node detects the existence of its neighbors, and then listens its medium to verify that there are not messages from one of its neighbors. In this example, *node 1* detects *node 2*, and due to the absence of messages in the medium, it sends a packet (*test message*) to *node 2*, which contains the maximum number of hops detected through the links with other neighbours. In this case, *node 1* has no more links, that is why in the packet the maximum number of detected hops through links different to the one with the *node 2* is *0*. *Node 2* has a table where it relates its neighbors (*node 1*) with the number of hops announced plus *1* (*0+1*). Finally *node 1* goes to a state that allows listening the medium again, in order to receive the response of *node 2*. The same process occurs in the other direction in case that both nodes start the process at the same time, or with a short difference of time.

Table 7.1: Information table of the node 2 at the network in the Figure 7.1.

ID	Address	Type	ECC	MAX_RECV	MAX2SEND
1	1.0.1.1	1	0	1	0

After processing the received packet, *node 2* sends a message (*accept message*) to *node 1* with the maximum number of hops detected by the other links, when *node 1* receives this message, stores the data in its table as *node 2* did. Now both nodes have the data from its neighbours, and have the ability to compute its eccentricity, so the next step is the evaluation of the vector that stores the number of hops on the table (MAX_RECV); the maximum of these values is the eccentricity of node, in this case, the eccentricity of each node is *1*. These new data is sent to neighbours by an *update message*. The exchange of messages is shown in Figure 7.2.

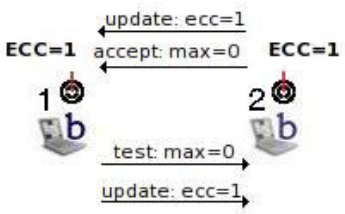

Figure 7.2: Message exchange in a network composed by two nodes.

Completing the process explained above, nodes listen the medium and if not more messages arrive or no new neighbors exist, then each node evaluates its possibility of being the*leader* of the network. In this case, no more packets that imply changes in the table will arrive, so each node verifies the information contained in its table to see if it is updated (see section 7.3.4) and proceed to compare its eccentricity with the eccentricity of its neighbors. The node with lower eccentricity can declare itself as the *leader* of the network, and communicates this event to all its neighbors using a *leader message*, but if the node detects that it has a higher eccentricity than some of its neighbors it must wait the arriving of a *leader message*. In some cases, the node eccentricity equals the eccentricity of one of its neighbors, so the *leader* position will be taken by the node with the higher ID between both of them. The *leader* in the Figure 7.2 will be *node 2*.

A net with two nodes is very simple, but with bigger networks others message dynamics can be seen. For example, after the sent or arriving of an *accept message* the involved nodes will evaluate if their eccentricity have changed due to the new data stored in the "MAX_RECV" vector, and will calculate the component values of the vector "MAX2SEND". If the eccentricity of the node has already changed then an *update message* will be sent to every neighbor, but if the "MAX2SEND" value of one of its neighbors has been updated then it will send an *update message* to the related neighbor. In contrast to the *update messages* that are only sent to some nodes, the *leader messages* must be spread through the entire network, in order to inform all the nodes about the identity of the *leader*.

In a more complex topology, the eccentricity and Leader will be calculated as shown in the Figure 7.3. The numbers that are located at each edge of the links indicate the number of hops to be announced by the neighbor plus one.

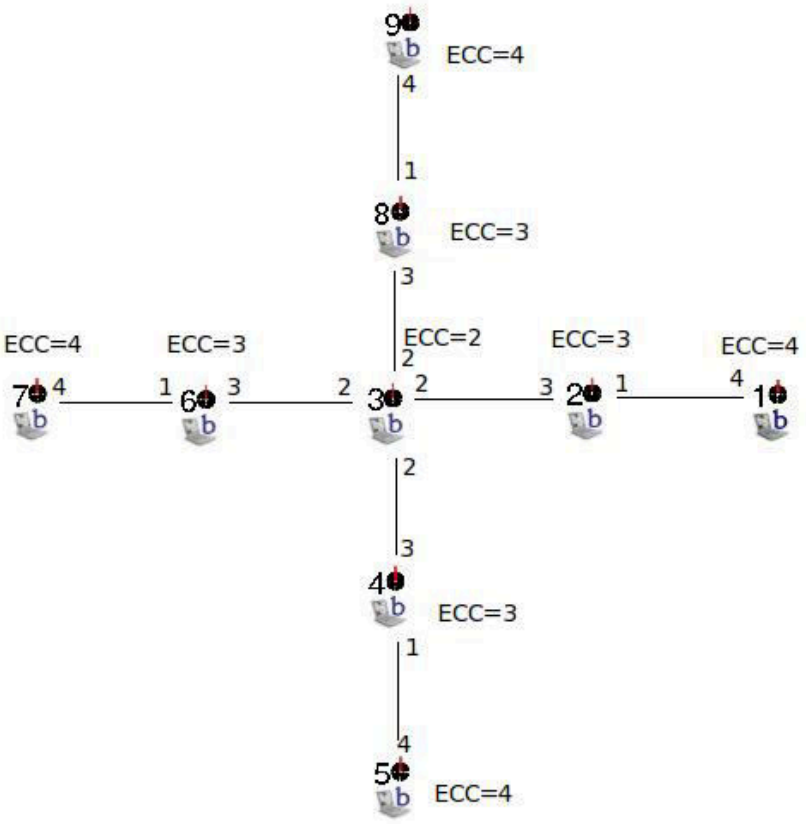

Figure 7.3: Calculation of the eccentricity of a network, where the Leader will be *node 3*.

Following the previous description, on a singleton graph, the unique node must realize that it does not have neighbors and then declare itself as the Leader of the network.

After this introduction into the operation of the algorithm, the message types used and details of specific functions are given.

7.3.2 Message Types

The algorithm uses four (4) types of messages, each one has its own responsibility into the process:

Test message

The *test message* has the objective to start the establishment of a link in order to exchange information that is used to calculate the eccentricity of each node. These messages are sent only to those neighbors with which there has not existed previously an exchange of messages related to the process of calculating the eccentricity, or those nodes that has not replied to a

previous *test message* and are still in the neighborhood.

The structure of a *test message* is shown in Table 7.2.

Table 7.2: Structure of a *test message*.

Node ID	Message type	Eccentricity	Max advised	Leader ID	Repetition indicator
int	unsigned int	unsigned int	unsigned int	unsigned int	unsigned int

For this message, the Message Type field has *1* as value.

Accept message

The *accept message* is used to confirm and accept the establishment of a link. This message is sent to those neighbors that have sent a *test message*. If an *accept message* is lost in the medium, a new *test message* will arrive, so the node will reply with an *accept message* indicating that it has been already accepted.

Its structure is the same as the shown in the Table 7.2, but with *2* as Message Type value.

Update message

The *update messages* are used to send updates of some events like merge or separation in the network, changes in the eccentricity or in some component of the vector "MAX2SEND". These messages are only sent and accepted if the evaluation of the conditions of the table indicates that an update is necessary for some of the neighbors.

The structure of an *update message* is the same of the *accept* and *test messages* (see Table 7.2), but the *repetition indicator* only has meaning for the *update message*. When the value of the *repetition indicator* is set in *1* is because the source of this has detected that its own table is outdated, so an *update message* will be sent to the node which the message came from. When an *update message* arrives and the value of the *repetition indicator* field is 0, the action to follow with the process changes according to the evaluation of the other fields:

- If the field *leader ID* announces that the *leader* of the network is the same node that has received the message, and the evaluation of the field *max advised* or *eccentricity* indicates that there are one or more outdated data, the current *leader* registered in the node is erased, and the node proceed to send the updates to all the neighbours that require the updating.
- If the field *leader ID* announces that the *leader* of the network is different to the currently registered in the node, the information of the table is updated accordingly to the message, and the node sends *leader messages* to the neighbours that can receive that kind of message from this node announcing about the change of leader.

Leader message

This message can advertise the existence of the network *leader*. It passes from the nodes with lower eccentricity to the nodes with higher eccentricity, and travels through the nodes until cover the entire network. The source of this kind of message is usually the *leader*. The *leader* transmits a *leader message* after discovering that it is the new *leader*, and also during the time in which there are no new events on the network, securing that all the nodes know which the *leader* is. When a *leader message* arrives to a node, it is forwarded to all the neighbors with higher eccentricity or equal eccentricity and lower ID, so it will not go back to the node that sent the message. This message is also generated after the arrival of an *update message* that reports a different *leader*.

The structure of a *leader message* follows the one that is shown in Table 7.3.

Table 7.3: Structure of a Leader Message.

Node ID	Message type	Eccentricity	Maintenance indicator	Leader ID
int	unsigned int	unsigned int	unsigned int	unsigned int

An event will always cause the updating of data in at least one of the nodes in the network. Events like the establishment or ending of a link with a node, are going to cause an *update message* informing the *leader* of the net must be deleted, and a new *leader* calculation must be done. But sometimes due to the lost of packages, a node can still have outdated information, so it may send this information to its neighbors. In order to avoid the spreading of false information, when a node discovers that it will be the *leader* of the network and will send this information the first time, it sets the *maintenance indicator* in *0*, but if the node had sent this information previously the *maintenance indicator* will be set in *1*. Thus, when a *leader message* arrives and the source meets the conditions of having a lower eccentricity, or equal eccentricity and higher ID, then the destination node will verify the *leader* that is being announced, if the *leader* is different from the current *leader*, then the new information is taken, but if the announced leader is the same as the leader who was previously deleted the information is only taken into account if the *maintenance indicator* has value *0*.

7.3.3 Type of nodes

According to the state of the links, the nodes have been categorized as *Accepted*, *Basic* and *Tested*. *Basic* nodes are those nodes that have been recently discovered and not message has been sent to it. *Tested* nodes are nodes that have been sent a *test message* and from which an *accept* message is required. For its part, *accepted* nodes are the ones which have already established a connection due to the exchange of *test* and *accept messages*.

The classification of the nodes depending on the type of messages exchanged is important because it indicates whether it is valid to receive the information sent by a node, and to control that all the neighbors have received the information, ensuring that the nodes will converge in the leadership election.

7.3.4 How does every node know that the eccentricity information is updated?

Proper operation for selecting the *leader* requires that the information stored in the tables are updated. With this in mind, the algorithm has been given the ability to discern whether the node and its neighbors have the information that corresponds to the current state of the network.

Supose the network of Figure 7.3, each node has a table to store information of its neighbors. *Node 4* has the information shown in Table 7.4.

Table 7.4: Information table of the *node 4* at the network of the Figure 7.3.

ID	Address	Type	ECC	MAX_RECV	MAX2SEND
5	1.0.1.5	1	4	1	3
3	1.0.1.3	1	2	3	1

As stated in table 7.4, *node 4* has two neighbours, and with both of them it has active links (indicated because the Type field is *1*). *Node 5* has advertised that its eccentricity is *4*, and through it *node 4* can reach zero nodes, so it save *0+1* as the maximum received. On the other hand, *node 3* has sent some messages indicating that its eccentricity is *2*, and that the number of nodes that *node 4* can reach through it is *2*, so it stores *2+1* as the maximum received from *node 3*. Based on the MAX_RECV vector *node 4* can make the calculation of what its eccentricity is. The eccentricity of *node 4* is $max\{max\_recv[node5], max\_recv[node3]\} = max\{1, 3\} = 3$. The maximum number of nodes that *node 5* can reach through *node 4* (MAX2SEND to *node 5*), is the maximum of the MAX_RECV vector without taking into account the link with *node 5*. The same process is used to calculate this value for *node 3*. With this operations and data, the following conditions are verified to ensure that the data of the table correspond to the current state of the network:

1. The difference between the own eccentricity, and the eccentricity advertised by a neighbour is no bigger than *1*.

2. The eccentricity advertised by a neighbour must be:

   - $ecc[neighbor(i)] = max\_recv[neighbor(i)] - 1$ **if** $(max\_recv[neighbor(i)] - 1) \geqslant (max2send[neighbo(i)] + 1)$
     This applies to the neighbours with lower eccentricity.
   - $ecc[neighbor(i)] = max2send[neighbor(i)] + 1$ **if** $(max\_recv[neighbor(i)] - 1) \leqslant (max2send[neighbor(i)] + 1)$
     This applies to the neighbours with larger or equal eccentricity.
   - The number of neighbours that have lower eccentricity is not larger than 1. Because of the procedure for calculating the eccentricity of every node, a node can

only have a neighbour with lower eccentricity or a neighbour with the same eccentricity.

- The own eccentricity is the maximum of the components of the MAX_RECV vector.

If the above conditions are met, then it can be ensured that data are updated and the choice of *leader* can continue. Each node checks its own table, and individually begins to verify the conditions to be or not the *leader* of the network.

7.3.5 How does a node realize that it can be the Leader of its network?

After a node confirms that its table is updated, it checks if it is the *leader* of the network. To be the *leader* of the network a node must:

1. Have lower eccentricity than all its neighbours.
2. If its eccentricity is equal to the eccentricity of one of its neighbours, its own ID must be higher. It is important to say that the node ID is a unique value that identifies it in the network. For example, the node addresses calculated with the method mentioned in the Section 1.3.1 at page 12 can be used, due to the fact that this process guaranties the uniqueness of the address.
3. Have successfully established communication with all its neighbours, i.e., all the neighbors have been marked as accepted.
4. Not to be a border node with eccentricity bigger than *1*. The only possibility for a node that is located at the edge of the network to be the *leader* is if the network is composed of at most two nodes.

In the case that the node detects that it is alone in its network, it declares itself as the *leader*.

7.3.6 Under what topology changes the Leader is reseted and a new Leader is selected?

In order to avoid false information, the *leader* of the network is reseted under some changes of topology. The network *leader* will be reseted in a node if one of the following occurs:

- The node is currently the *leader* of the network, but the conditions have change and now there is a node with a lower eccentricity, equal eccentricity and higher ID, or its

table is outdated. The node also deletes itself as *leader*, if it is located in the edge of the network and its eccentricity is bigger than *1*.

- There is a *leader* in the network, but the node has discovered a new node, or there is a node marked as *tested* in its table.
- The node detects that it has been separated from some of its neighbours.
- The node receives a test message (a neighbour discovered it).
- The node detects that the difference between its eccentricity and the eccentricity informed by one of its neighbours is bigger than *1*, or some data in its table is outdated, see section 7.3.4.

The reset of the network *leader* leaves place for the selection of a new *leader* that obeys to the topology conditions produced by the nodes movement.

7.3.7 How do two networks merge?

The partition and the merging of networks is a consequence of the topology dynamics, and must be addressed if the objective is to obtain an algorithm that works under free movement conditions.

As has been explained, if a network is composed by unconnected subnetworks, each subsystem will have its own *leader*. When the subsystems are too close and a link is established among them, then a unique *leader* must be chosen. In this way, at least one node in each subsystem must detect a new neighbour that has a *leader* different from its own *leader*. So, the links will be established and both of them will reset their *leader*, then they will inform their neighbours about this event and finally the new merged network will start the selection of a *leader*.

7.3.8 What does happen when a network is divided in two or more pieces?

Following the conditions to reset a *leader*, when a node loses contact with one of its neighbours, the *leader* is reseted, and if the network is divided in two or more pieces, each piece selects a new *leader* obeying to the new conditions. The separation is informed to the neighbours using *update messages*.

7.3.9 Other functions for the eccentricity

As has been explained through that chapter, the eccentricity is used to calculate the central node of the network, but after it is achieved, the eccentricity will be used in the communi-

cation process to transmit the messages related to the IDS and its sensors, so the data will travel from the nodes with higher eccentricity to the nodes with lower eccentricity. Thus, the time spent to reach the node which analyses the global data is minimized under the assumptions mentioned in the section 7.1.

7.4 Verification and Validation of the Algorithm

The process of verification and validation is presented in the section 8.3.

7.5 Complexity Analysis

Although the time complexity is very important for the purpose of know how demanding an algorithm is for the machine, when the algorithm implies the sending of messages among nodes, is very important to calculate the message complexity, which makes reference to the quantity of messages that need to be exchanged in order to meet the objectives, and in this case to converge to the same result in each node.

To analyse the messages complexity ten examples are analysed, and then the results are extended to a network composed of n nodes.

For the next calculation it is assumed that there are no losses of messages [27] and every sent message is answered immediately.

First, the whole process is explained in detail for a pair of nodes, say *node 1* and *node 2*, where *node 1* starts the process a few seconds before *node 2*.

The process in *node 1* is the following:

1. *Node 1* initializes the process by entering into a state called *init*. In this state no messages are sent.
2. Then, the process continues with the *state_testing* in which the node goes over the list of neighbors until find a basic node. In the current case, due to the fact that *node 2* is the unique neighbor of *node 1* and has not established any link, it will be the addressee of a *test message*.
3. After sending the message, *node 1* waits (for a limited quantity of time) a response in the *state_listening*, once a message arrives the next state is called.
4. In the *state_msg_evaluation* the incoming message is read. Assuming that the received message is of type *accept* the next state is invoked.
5. In the *state_updating* the eccentricity and the MAX2SEND is calculated, then all the neighbours must be informed about the new established link, the new eccentricity and

the correspondent MAX2SEND. As the eccentricity of *node 1* has changed, it sends an *update message* to *node 2*.

6. After completing step 5, the process will go to step 2 to verify if the node can be the *leader* of the network, but it will realize that the eccentricity of *node 2* is not updated, so it can not be the *leader*. *Node 1* discovers that it has not more neighbors that are prone to establish links. Hence, the process will go from *state_testing* to *state_listening* until the arriving of some message.

7. When the *update message* from *node 2* arrives, *node 1* will have its table completely updated, so it will realize that it cannot be the *leader* of the network due to the fact that *node 2* has the same eccentricity and a higher ID.

For its part, the *node 2* will execute the following process:

1. *Node 2* has autonomy over the time to start the execution of the process, in this case the assumption made is that it starts after *node 1* has started, but is important to clarify that the start time of *node 1* and *node 2* are completely independent. *Node 2* initializes the process in the *state_init*, and then it invokes to *state_testing* to verify its neighborhood.

2. In the *state_testing*, the node has the capability to listen the medium before start sending messages to its neighbors. Due to the fact that by this time *node 1* had sent a *test message*, it will be listened by *node 2*. To analyze the message, the next state is called.

3. In the *state_msg_evaluation* the node interprets the message as type *test* and sends an *accept message* to *node 1*.

4. Given the new accepted neighbor, it is necessary to inform the other members of the network about this event. So the *state_updating* is executed, and there it will send an *update message* to all nodes that are detected that have their table outdated including *node 1*.

5. The process goes back to *state_testing* and *node 2* listens the *update message* from *node 1*, updates its table and goes to *state_updating*, but so it returns to *state_testing*.

6. In the *state_testing*, *node 2* realizes that it meets all the conditions to become the *leader* of the network, and since there are no messages to listen or basic nodes, it proceeds to send a *leader message* to *node 1*.

In the Figure 7.4 the exchange of message between Node 1 and Node 2 is clarified.

In practice, both nodes can start the process with a little difference of time between them. The worst case will be that both nodes send *test* and *accept messages* at the same time. The

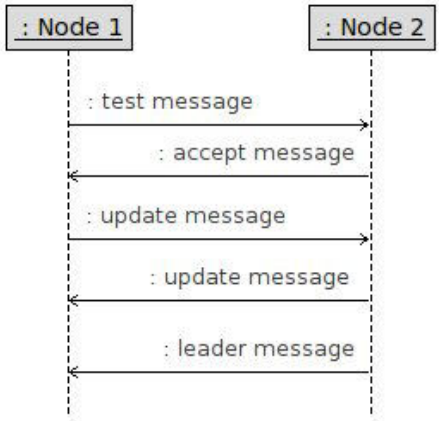

Figure 7.4: Sequence diagram of the exchange of message among nodes in the process of link establishment between *node 1* and *node 2*.

update messages are generated by the node that has suffered some change, so the worst case for the process of *updating* will be forward of an *update message* asking for a response when it has detected that the neighbor has not sent an update after sending of a message that informs the neighbor about a change in the *max2send*, and implying a change in the eccentricity of the addressee. Thus, from now each *update message* that updates the *max2send* and implies a change in the eccentricity of the addressee is duplicated.

Now the process for a link establishment is clear, and the message complexity can be calculated:

For two nodes:

- 2 *test messages* are generated (in the worst case).
- 2 *accept messages* are generated (in the worst case).
- 2 *update messages* are generated to inform about the change in the value of the *eccentricity*.
- 1 *leader message* is generated.

For three nodes:

When a third node arrives to the previous net the exchange of messages will be as the Figure 7.5 shows:

In summary:

- 2 *test messages* are generated (in the worst case).

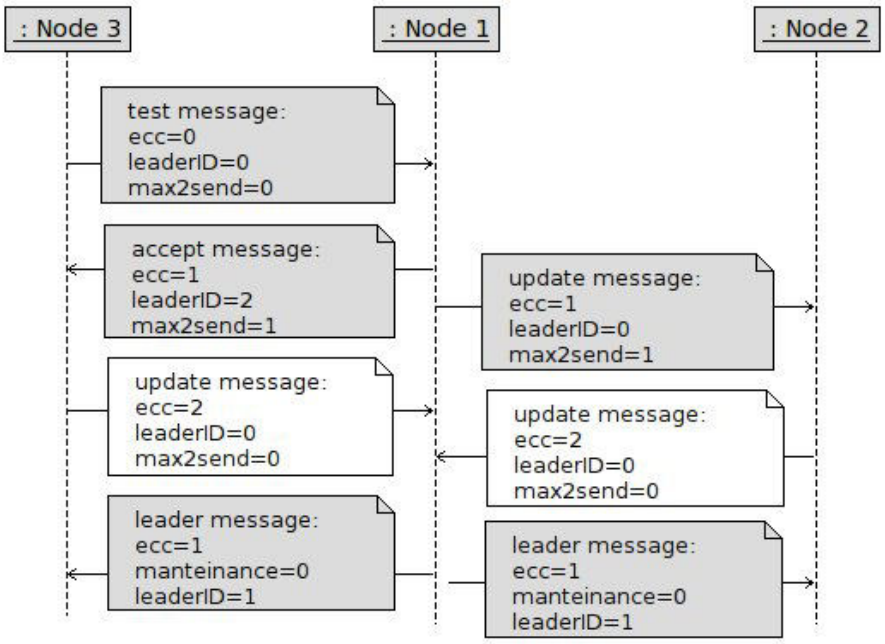

Figure 7.5: Sequence diagram of the exchange of message among nodes in the process of a second link establishment when *node 3* joins a net composed of two nodes.

- 2 *accept messages* are generated (in the worst case).
- 4 *update messages* are generated. 2 in order to inform the change of *eccentricity* (see the white messages), 1 to send information about the change in the *max2send* and 1 as duplicated[1].
- 2 *leader messages* are generated.

For four nodes:

When a fourth node arrives to the previous net the exchange of messages will be as the Figure 7.6 shows:

In summary:

- 2 *test messages* are generated (in the worst case).
- 2 *accept messages* are generated (in the worst case).

[1]Remember that this has been duplicated having into account the worst case of the *update messages*. See page 49. The number of duplicated messages equals the number of *update messages* that changes the eccentricity of the addressee

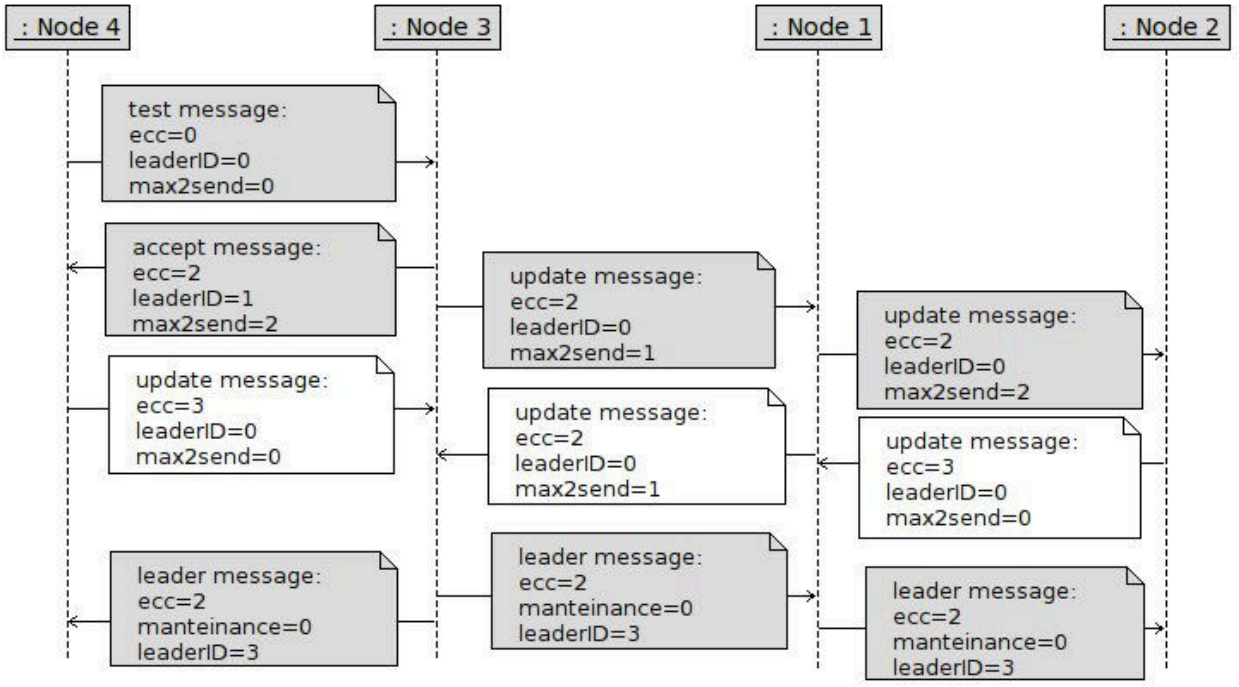

Figure 7.6: Sequence diagram of the exchange of message among nodes in the process of a third link establishment when *node 4* joins a net composed of three nodes.

- 7 *update messages* are generated. 3 are only used to inform the change of *eccentricity*, 2 to send mixed information about the change in the *max2send* and in the *eccentricity*, and the last 2^1 are duplicates.
- 3 *leader messages* are generated.

For five nodes:

When a fifth node arrives to the previous net the exchange of messages will be as the Figure 7.7 shows:

In summary:

- 2 *test messages* are generated (in the worst case).
- 2 *accept messages* are generated (in the worst case).
- 8 *update messages* are generated. 3 are only used to inform the change of *eccentricity*, 3 send mixed information about the change in the *max2send* and in the *eccentricity*, and 2^1 are duplicates.
- 4 *leader messages* are generated.

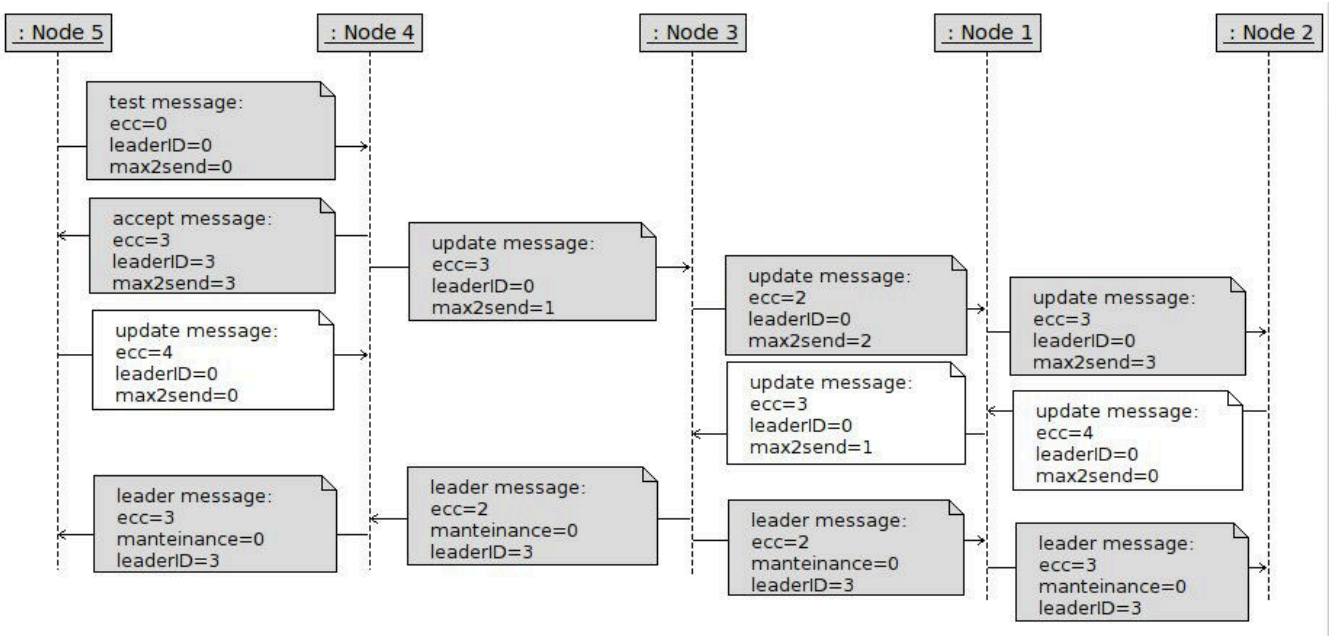

Figure 7.7: Sequence diagram of the exchange of message among nodes in the process of a fourth link establishment when *node 5* joins a net composed of four nodes.

For six nodes:

When a sixth node arrives to the previous net the exchange of messages will be as the Figure 7.8 shows:

In summary:

- 2 *test messages* are generated (in the worst case).
- 2 *accept messages* are generated (in the worst case).
- 11 *update messages* are generated. 4 are only used to inform the change of *eccentricity*, 4 send mixed information about the change in the *max2send* and in the *eccentricity*, and 3^1 are duplicates.
- 5 *leader messages* are generated.

The messages of the six cases plus four more are summarized in the Table 7.5.

In the best case, the sending of *leader messages* will be produced only at the end of the establishment of all the possible links. In the worst case, after the establishment of a link there will be enough time to calculate the *leader* of the network and send the *leader messages*, so when a new node joins the net, the former *leader* should be deleted and a new one must

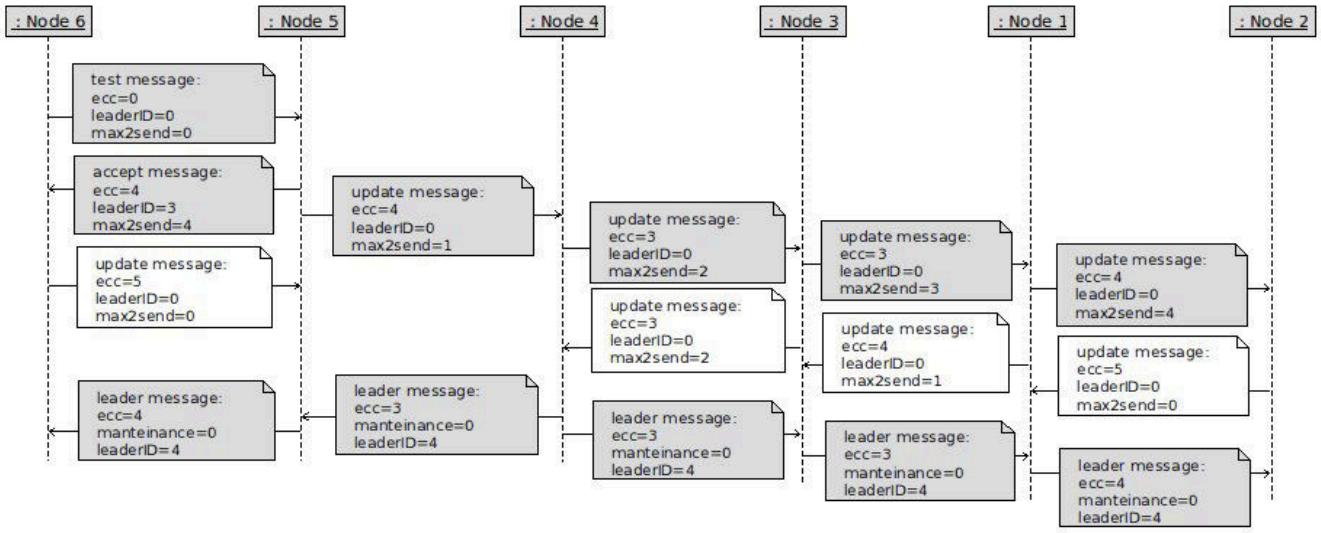

Figure 7.8: Sequence diagram of the exchange of message among nodes in the process of a fifth link establishment when *node 6* joins a net composed of five nodes.

be calculated. To be consistent in the message complexity, only the worst cases for each type of messages are taken into account.

First of all, the total number of messages is calculated. The total quantity of messages necessary for the establishment of the links and the leader election in a net of n nodes is represented by the Equation (7.1).

$$TotalMessages(n) = \sum_{i=1}^{n} (T_i + A_i + UE_i + UM_i + UD_i + L_i) \tag{7.1}$$

Where:

T represents the *test messages*,
A represents the *accept messages*,
UE represents the *update messages* that informs the change in the *eccentricity*,
UM represents the *update messages* that informs the change in the *max2send* and the change in the *eccentricity*,
UD represents the duplicated *update messages*,
L represents the *leader messages*.

For the *test messages*:

Table 7.5: Messages for the process of establishment of links and leader calculation.

#_of_nodes	Test	Accept	Update:Ecc	Update:Mixed	Update:Duplicated	Leader
1	0	0	0	0	0	0
2	2	2	2	0	0	1
3	2	2	2	1	1	2
4	2	2	3	2	2	3
5	2	2	3	3	2	4
6	2	2	4	4	3	5
7	2	2	4	5	3	6
8	2	2	5	6	4	7
9	2	2	5	7	4	8
10	2	2	6	8	5	9

$$\sum_{i=1}^{n} T_i = 2\,(n-1) \tag{7.2}$$

For the *accept messages*:

$$\sum_{i=1}^{n} A_i = 2\,(n-1) \tag{7.3}$$

For the *update messages*, which information only updates the *eccentricity*:

$$\sum_{i=1}^{n} UE_i = \begin{cases} 2\sum_{i=2}^{\frac{n+1}{2}} i = \dfrac{n^2+4n-5}{4} & \text{, if } n \text{ is odd} \\ 2\sum_{i=2}^{\frac{n}{2}} i + \dfrac{n}{2} + 1 = \dfrac{n^2+4n-4}{4} & \text{, if } n \text{ is even} \end{cases} \tag{7.4}$$

For the *update messages* that informs at the same time the change in the *eccentricity* and in the *max2send*:

$$\sum_{i=1}^{n} UM_i = \sum_{i=3}^{n} (i-2) = \frac{n^2-3n+2}{2} \tag{7.5}$$

The *update messages* that are duplicated in the worst case are:

$$\sum_{i=1}^{n} UD_i = \begin{cases} 2\sum_{i=4}^{\frac{n+3}{2}} (i-2) + 1 = \dfrac{n^2-5}{2} & \text{, if } n \text{ is odd} \\ 2\sum_{i=4}^{\frac{n+2}{2}} (i-2) + \dfrac{n}{2} + 1 = \dfrac{n^2-4}{4} & \text{, if } n \text{ is even} \end{cases} \tag{7.6}$$

And finally the total number of *leader messages* in the worst case is:

$$\sum_{i=1}^{n} L_i = \sum_{i=2}^{n} (i-1) = \frac{n^2-n}{2} \tag{7.7}$$

Now, replacing equation (7.2), (7.3), (7.4), (7.5), (7.6) and (7.7) in (7.1) the result is:

$$TotalMessages(n) = \begin{cases} \dfrac{3n^2+6n-11}{2} & \text{, if } n \text{ is odd} \\ \dfrac{3n^2+6n-10}{2} & \text{, if } n \text{ is even} \end{cases} \tag{7.8}$$

The result in the Equation (7.8) indicates that the proposed algorithm has a message complexity of $O(n^2)$ in the worst case.

Chapter 8

Extension of the Algorithm for the Topology Establishment and Avoidance of Loops

In Chapter 7 the algorithm that was presented solves the election of the *leader* assuming the existence of a *Topology Control Protocol*. In this Chapter, the algorithm is extended in order to guarantee its correct operation in nets where a *Topology Control Protocol* has not been implemented. Hence, the algorithm accomplishes also *Topology Control Protocol* functions such us the establishment and selection of communication links and the elimination of loops. In the previous chapter, with the *Topology Control Protocol*, the *leader* node was located at the point of eccentricity of the topology established by it, but with the absence of the protocol, the *leader* node is approximately located at the point of eccentricity of the physical topology of the network.

Other assumptions obey to the stated in Section 7.1.

8.1 Design and definition of the Algorithm

The first responsibility of this extension is the elimination of physical loops among the nodes, then the improvement of the eccentricity of each node is necessary in order to obtain a net where all the nodes have the lowest possible eccentricity, this is what makes possible to situate the *leader* node approximately at the point of physical eccentricity of the network.

8.1.1 A general description of the algorithm

Unlike the previous approach, with this extension once a neighbour B has been discovered *node* A sends it a *test message*, when B receives the message and A is going to be accepted,

B initializes a process to evaluate and avoid the existence of loops. This process is also initialized by *node A* after the arriving of the *accept message* that comes from node *B*. If a loop is detected by a node, the link that is involved in the loop is discarded by marking the related neighbour as *rejected*. Once the network reaches the stability in the eccentricity of all the nodes, the leader is elected.

8.1.2 Type of messages

This extension introduces the existence of two more messages:

Init message

Init messages are used to evaluate the presence of loops in the network. This process is initiated after the reception of a *test message* when the source of the message is going to be accepted, also when an *accept message* arrives or due to the reception of a *leader message* in a node whose ID is the same that the leaderID informed by the message. This message has the structure shown in Table 8.1.

Table 8.1: Structure of an Init Message.

Node ID	Message type	Source
int	unsigned int	unsigned int

This message is created by a node that is in the process of establish a new link, so the node ID is registered in the *source* field and it is only changed if arrives to a node that has a lower ID. Each time the message arrives to a new node it is forwarded and the Node ID field is updated according to the ID of the node that forwards the message. This message is recognised as having *5* as Message Type.

Test message

In addition to the functions mentioned before in the section 7.3.2, here the *test messages* are also used to test if a node that has been previously rejected can be accepted under the current topology conditions.

Reject message

When this kind of message is received in a node, it is because a neighbour has the objective to indicate that it wants to avoid the use direct communication at least until a new state of topology takes place.

This message is sent to the neighbour that does not offer a good position when a node discovers a possibility to improve its current eccentricity, or when the establishment of a link with a neighbour forms a loop. It is also used when a message has been received from a node that has been previously rejected.

The structure of the *reject message* is similar to the *test message* structure, but the

Message Type value is *0*; see Table 7.2.

8.1.3 Type of nodes

Due to the new conditions imposed by the loop avoidance, some links must not be used, to secure that those links are not used, the nodes are also marked as *Rejected*, and thus the nodes can exchange messages with all their neighbours that do not have been marked as rejected.

8.1.4 How does the algorithm avoid infinite eccentricities in the presence of physical topology cycles?

The way that the eccentricity is calculated, leaves the possibility that if there is a loop in the network, the eccentricity of the nodes increases without limit. Mid Detection [63] is a strategy used to avoid this kind of problems in MANETs, and consists on the detection of loops through the use of packets that have a field for the source address or ID, and when this kind of packet arrives to its destination the source field is verified and compared with the next hop ID, if it coincides then a loop is detected and the packet will not be send to the next hop.

Here, a *Mid Detection* implementation was done with a little modification, and was mixed with the change in the *source* field under certain conditions that are suggested by the *Opt-FloodMax* strategy in [58]. Each time that a new node arrives to the network or a new neighbor is discovered (say *node B*), a *test message* is sent. When the message is received in the destination it evaluates the information and marks *node A* as *rejected* or *accepted*. If *node A* is accepted, then an *init message* is sent by *node B*, in order to initialize the loop detection procedure.

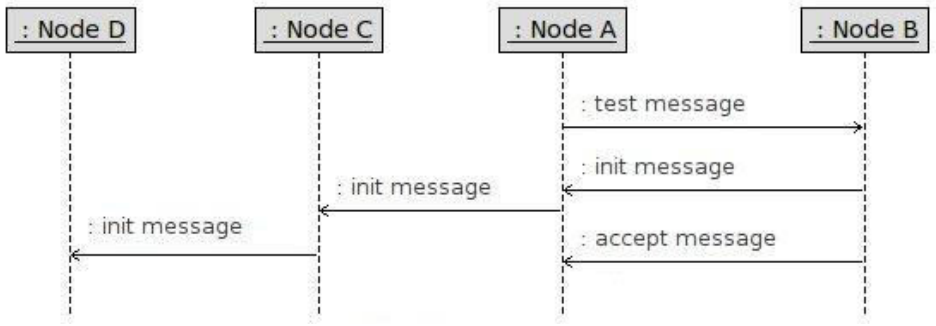

Figure 8.1: Exchange of *init message* among nodes in a network.

The *init messages* are forwarded each time they arrive to a node. In order to avoid multiple loop detection procedures that can generate network partitions, the message which

the source is the node with the lower ID in a path will be sent without changes in the source to all neighbours (the unique node that will not receive this message will be the node where it came from), but if the source of the message will be changed if it arrives to a node who has a lower ID. If an *init message* arrives to a node whose ID is equal to the source of the packet, a loop is detected and the packet is discarded, then the node that it came from is marked as rejected and will receive a message with this information. Thus, the loop is broken and the process to calculate the eccentricity will find a proper stability. See in Figure 8.1 the way that *init messages* are exchanged when a new neighbour is discovered.

When a link is broken, as described above, the condition to always preserve at least one connection is evaluated to prevent the isolation of the nodes.

Figure 8.2 shows how a loop is detected and how eccentricity is calculated.

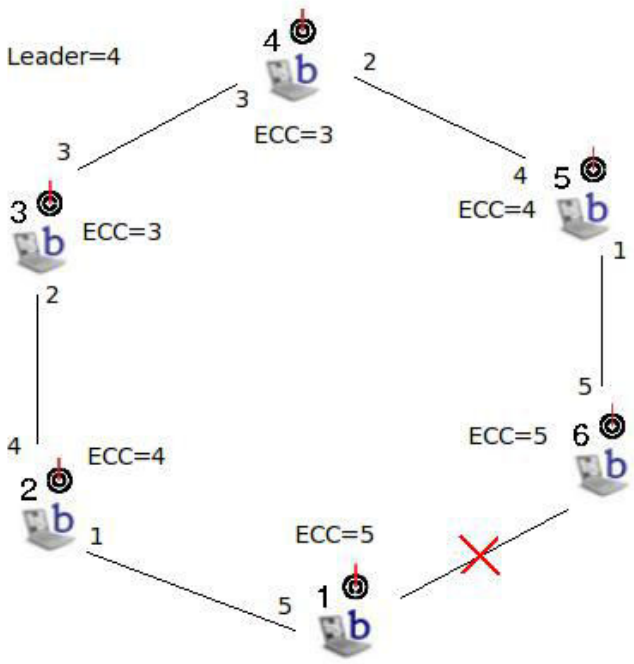

Figure 8.2: Calculation of the eccentricity of a network with loops.

8.1.5 How does a node improve its position?

As has been mentioned, one of the main features of MANETs is their dynamic topology, which continuously changes the network state. The movement cause the establishment or dissolution of links; this can imply for example, that some nodes get a better eccentricity using the new link instead of the old one. This problem is solved in this section.

Suppose the network of Figure 8.3.

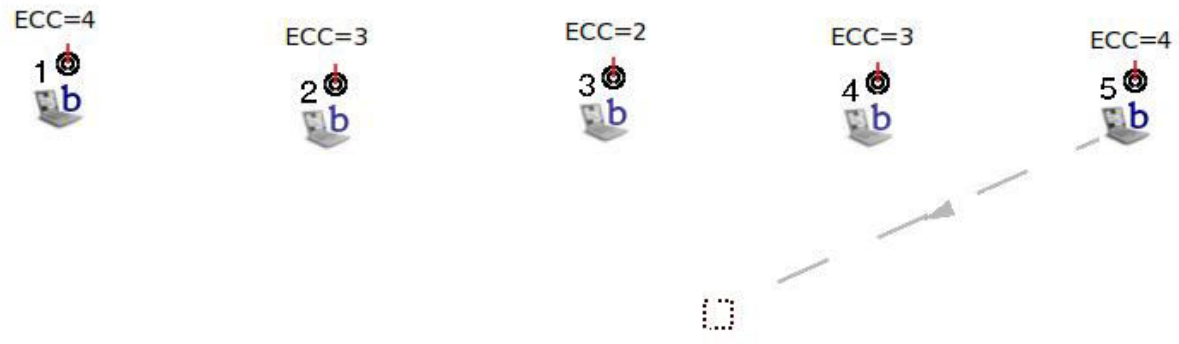

Figure 8.3: Movement of a node and the possibility of improve its eccentricity.

As can be seen, the eccentricity of *node 5* is initially *4*, and the *leader* of the network is *node 3*. With the movement of *node 5*, a new link between *node 5* and *node 3* is established. When *node 3* detects *node 5* sends a *test message*. So *node 5* discovers that *node 3* has the same *leader* but has a lower eccentricity. Then, *node 5* rejects (marks as inactive) to the first node that has higher eccentricity than its new neighbour, makes a loop test to *node 3* and establish an active link with it, and finally resets its *leader*. The reset of the *leader* is known by all the nodes of the network due to the *update messages*, which indicate to the network that a new *leader* must be found.

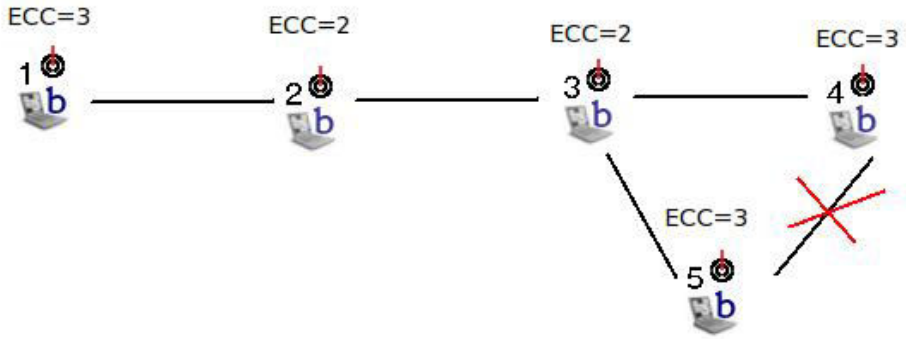

Figure 8.4: Movement of a node and the possibility of improve its eccentricity.

The process results in the network shown in Figure 8.4, where the *leader* is again *node 3*.

8.2 Limitations of the Algorithm

Some networks are composed of complex topologies where a node can be involved in nested loops. That type of networks means a problem to this algorithm. Figure 8.5 shows a network, where each node is involved in at least 4 loops.

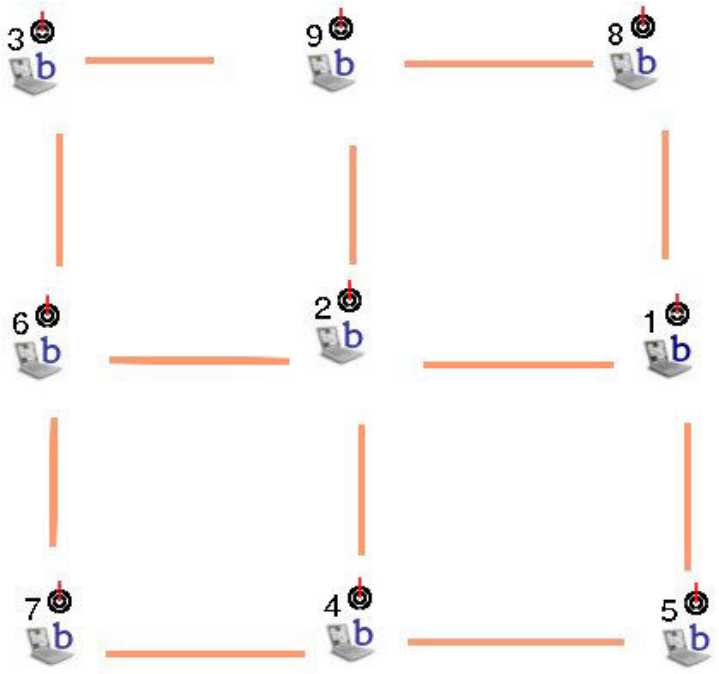

Figure 8.5: A network with nested loops.

This algorithm tests loops using a metric that is not related with topology, although if a topology metric, like node degree or the number of cycles of a node, is used it does not helps so much due to the fact that in some topologies (for example pyramid topologies) any node can be the centre of the network, so it is not a good solution.

8.3 Verification and Validation of the Algorithm

The specification, design, verification and validation of the algorithm were tasks that were implemented in an iterative manner. Thus, the first specification made was the related to the simplest function of the algorithm, and then a model of this was done using the Petri Net approach. The resulting net was analysed in order to ensure that the model cover all the possible events for the conditions under which the system could be found at some point, and also to evaluate the performance of properties like deadlock free and safeness. If the model showed that the specification had some problem, then it was fed back to find a solution to

reach the desired results. Once the obtained model was verified, the function was coded and validated using the simulation tool, if the codification or the results obtained showed some problem with the conditions and calculations, then the model was readjusted. If the desired results was obtained, then the next function was integrated to the previous model to be verified, and repeat the process until achieve the validation. This method was used for all functions, to accomplish the specification of the algorithm.

The construction of the model and its verification was realized on PIPE (Platform Independent Petri net Editor). The evaluation of the Petri Net behavioral and structural properties gives clues about the characteristics that the system has. Appealing to Murata [66], the behavioural properties are: reachability, boundedness, liveness, safeness, reversibility and home state, coverability, persistence, synchronic distance and fairness; and the structural properties are: structural liveness, controllability, conservativeness, repetitiveness, consistency, S- and T-invariants, structural B-fairness. Taking into account the characteristics of the transmission medium and the way that communication works, only some of these properties need to be met strictly.

PIPE is a powerful tool, that includes some others tools (like DNAmaca which is useful for the analysis of complex Timed Petri Nets) to improve the analysis. PIPE was chosen to model the algorithm because it can handle large nets without problems, simulating and analysing properties. In addition, it is an open source tool.

The analysis shown at Figure 8.6 corresponds to the evaluation made over the complete model of the algorithm for an isolated node.

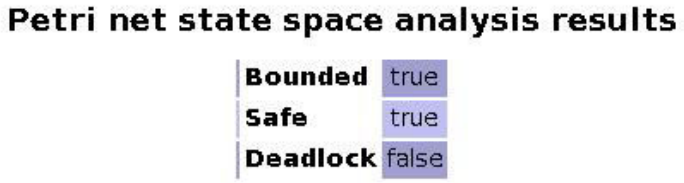

Figure 8.6: Space State analysis for an isolated node.

Figure 8.6 indicates: first, the model of the algorithm is bounded, so the algorithm does not have problems of overflow; second the Petri net is safe, this implies that the maximum number of tokens present in any place is 1, so the system can only execute the processing of one message at time [43] that makes sense according to the specification of the algorithm; and finally, the net does not have deadlocks, thus that the algorithm will continue working no matter in which state it is.

The classification analysis provided by PIPE gives the result presented in the Figure 8.7.

The model corresponds to a state machine and not to a marked graph, indicating that the algorithm does not have synchronization processes, however decisions can be found. The structure of the net corresponds to a Free Choice Net, and due to the fact that Free Choice

Figure 8.7: Classification analysis.

Nets are subnets of the Extended Free Choice Nets, then it is also an Extended Free Choice Net. The net also has the property of be a Simple and Extended Simple Petri net. See page 27.

The reachability tree has 27 nodes, which means that the total number of states or different markings reachable in the Petri net is 27, as shown in Figure 8.8. As can be seen, all the states have input and output arcs, except the state $S0$ that corresponds to the initial marking of the net. According to the definition of liveness (see page 30) this net is not alive, but it does not imply a problem because it is not necessary that the net go back to this state $S0$, which represents the initial state in which the nodes do not know their neighbourhood.

The blue nodes represent vanishing markings, that are markings that do not spend enough time to be visualized, the tangible markings that are represented by red nodes, corresponds to markings that are related with timed transitions [17].

After this analysis, the model was replicated, and the new model is composed by two nodes running the algorithm. Nodes are in the range of each other. The space state analysis result indicates that the net is bounded, safe and does not have deadlocks. In this case, the characteristics of boundedness and safeness are very important, because they also make reference to the transport medium, which can only manage one message at time, no matter the strategy that is being used (multichannel, or monochannel). The result of this analysis is shown in Figure 8.9, and the classification in Figure 8.10.

The net is no a *state machine* due to the model of the transmission medium which has, adds or transforms some transitions to have more than one input place (*synchronization*) or more than one output place. The *conflicts* that existed before were not affected by the medium model, so the net still being a not marked graph. The integration also implied the presence of *confusion*, so the net cannot be classified as *Free Choice*, *Extended Free Choice*, *Simple* or *Extended Simple*. *Confusion* implies the presence of *conflicts* in the net, but they are solved deterministically.

For the model of two nodes, it was not possible to construct the reachability graph, but it was verified through the use of simulation the firing of all the transitions.

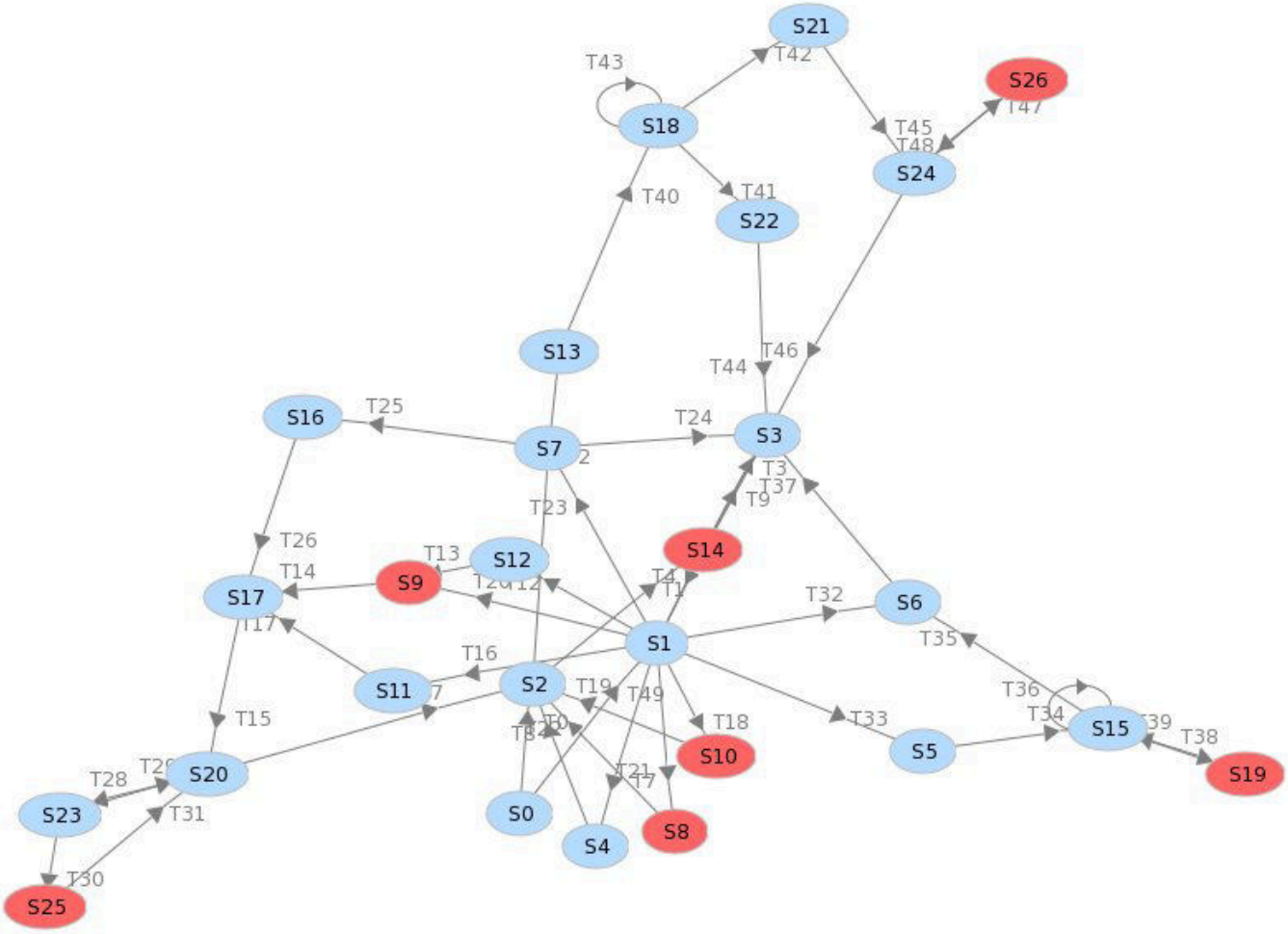

Figure 8.8: Reachability graph. Blue nodes are Vanishing nodes, and red nodes are Tangible nodes.

Petri net state space analysis results

Bounded	true
Safe	true
Deadlock	false

Figure 8.9: Space State analysis of the model for two nodes executing the algorithm

Although of the importance of all the properties of the Petri nets, the liveness and safeness are the more relevant in the study of the correctness of an algorithm where the communication is an important fact [66]. The analysis made to the model of the algorithm shows that effectively the algorithm is safe and bounded, but not alive for the reasons given before,

Figure 8.10: Classification of the model for two nodes executing the algorithm

however if the first state were eliminated of the model (because is no relevant to go back to this state) the net becomes alive. Thus, the requirements for the correctness of the algorithm are met.

8.4 Complexity Analysis

Here the complexity is calculated for those nets with loops, but not with nested loops, because although the algorithm can solve this topologies, it was not the goal of this thesis and would not give a good solution for such as cases.

Similar to the process followed in the section 7.5, some examples are presented and then their complexity is calculated. The examples start with two nodes due to the fact that if the net is composed of only one node, no messages will be exchanged. Under the assumption that nodes cannot appear or disappear suddenly in the net, the possibility that a new node can establish two links at the same time is low, but it has been considered as the worst case, thus, each time that a new node arrives to the net it will establish two new links conforming a loop, after the process of loop evaluation one of them will be dissolved.

Is important to have into account that after the establishment of the two links and until the dissolution of the loop, nodes will stay sending *update messages* obeying to the continuous change of the eccentricity. When the new topology is found, nodes will send *update messages* in order to obtain the final eccentricity that will allow the existence of the *leader*.

For two nodes:

- 2 *test messages* are generated (in the worst case).
- 2 *accept messages* are generated (in the worst case).
- 2 *update messages* are generated to inform about the change in the value of the *eccentricity*.

- 1 *init message*.
- 1 *leader message* is generated.

For three nodes:

Suppose that a net with two nodes has found its *leader*, and a third node arrives. Given to the fact that the node that sends the *accept message* is the one that initializes the loop evaluation, the largest number of messages occurs when *node 3* is the one that accepts to *node 1* and *node 2*, so it will be considered as the worst case.

Figure 8.11 shows the messages that are exchanged among the nodes in order to break the loop and calculate the eccentricity.

The calculation of the total number of messages requires to find a pattern of the way that messages are generated. To facilitate this process, the messages that are produced before and after the breaking of the loop are identified and separated.

In the Figures that will be showed in the rest of this section, messages are associated with colours in the following way: *test* and *accept messages* are drawn in black, *init messages* are in dark gray, *update messages* that inform a change in the *max2send* or change in the *max2send* and the *eccentricity* are blue-green (teal), and the *update messages* that only inform about a change in the *eccentricity* field are dark blue.

In summary for the three nodes, the messages before and during the breaking of the loop are:

- 4 *test messages* are generated (in the worst case).
- 4 *accept messages* are generated (in the worst case).
- 8 *init messages*.
- 8 *update messages* are generated, 4 announce a change in the *max2send*, and 4 more are considered under the conditions of the worst case (see the previous chapter).
- 4 *update messages* that inform a change in the *eccentricity*.

After the breaking of the loop the messages are:

- 1 *init message* is generated.
- 8 *update messages* are generated. 4 announce a change in the *max2send*, and 4 more are considered under the conditions of the worst case.
- 2 *update messages* that inform a change in the *eccentricity*.
- 2 *leader messages* are generated.

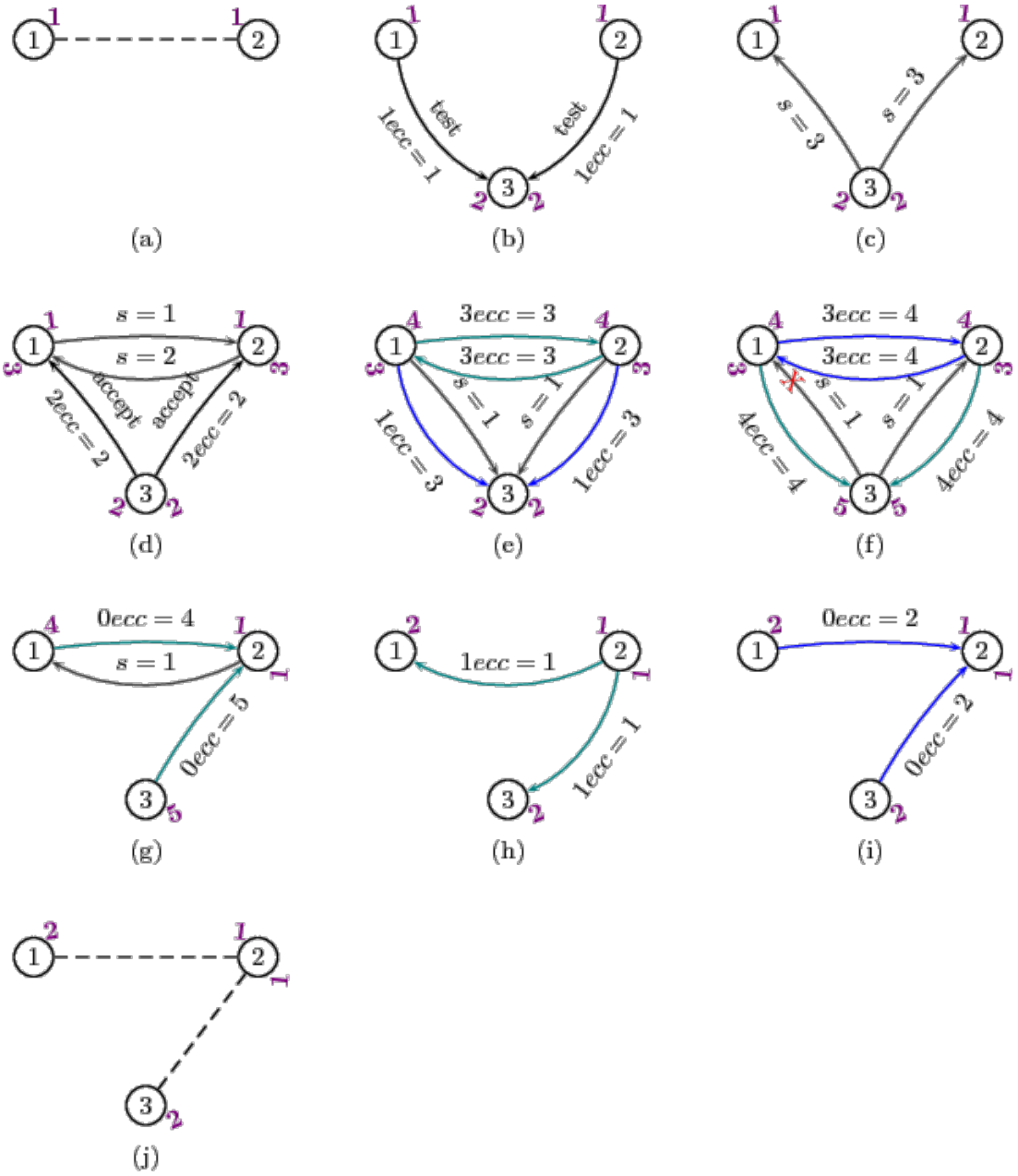

Figure 8.11: Message exchange in a net with three nodes.

For four nodes:

When a fourth node arrives to the previous net, the exchange of messages will be as Figure 8.12 shows:

In summary for the four nodes, the messages before and during the breaking of the loop are:

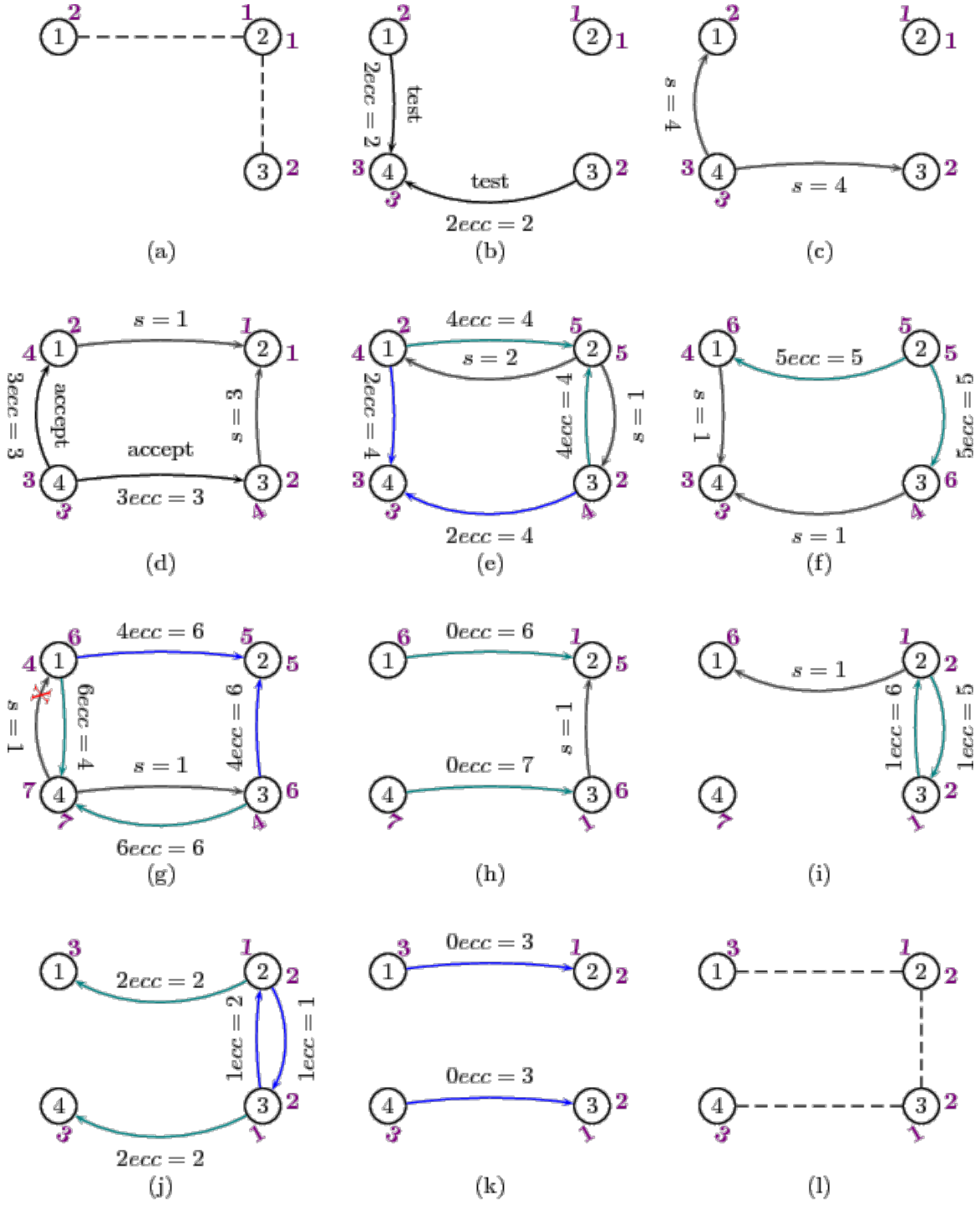

Figure 8.12: Message exchange in a net with four nodes.

- 4 *test messages* are generated (in the worst case).
- 4 *accept messages* are generated (in the worst case).

- 10 *init messages*.
- 12 *update messages* are generated, 6 announce a change in the *max2send*, and 6 more are considered under the conditions of the worst case (see the previous chapter).
- 4 *update messages* that inform a change in the *eccentricity*.

After the breaking of the loop the messages are:

- 2 *init messages* are generated.
- 12 *update messages* are generated. 6 announce a change in the *max2send*, and 6 more are considered under the conditions of the worst case.
- 4 *update messages* that inform a change in the *eccentricity*.
- 3 *leader messages* are generated.

For five nodes:

When a fifth node arrives to the previous net, the exchange of messages will be as Figure 8.13 and 8.14 show:

In summary for the five nodes, the messages before and during the breaking of the loop are:

- 4 *test messages* are generated (in the worst case).
- 4 *accept messages* are generated (in the worst case).
- 12 *init messages*.
- 16 *update messages* are generated, 8 announce a change in the *max2send*, and 8 more are considered under the conditions of the worst case (see the previous chapter).
- 8 *update messages* that inform a change in the *eccentricity*.

After the breaking of the loop the messages are:

- 3 *init messages* are generated.
- 16 *update messages* are generated. 6 announce a change in the *max2send*, and 6 more are considered under the conditions of the worst case.
- 4 *update messages* that inform a change in the *eccentricity*.
- 4 *leader messages* are generated.

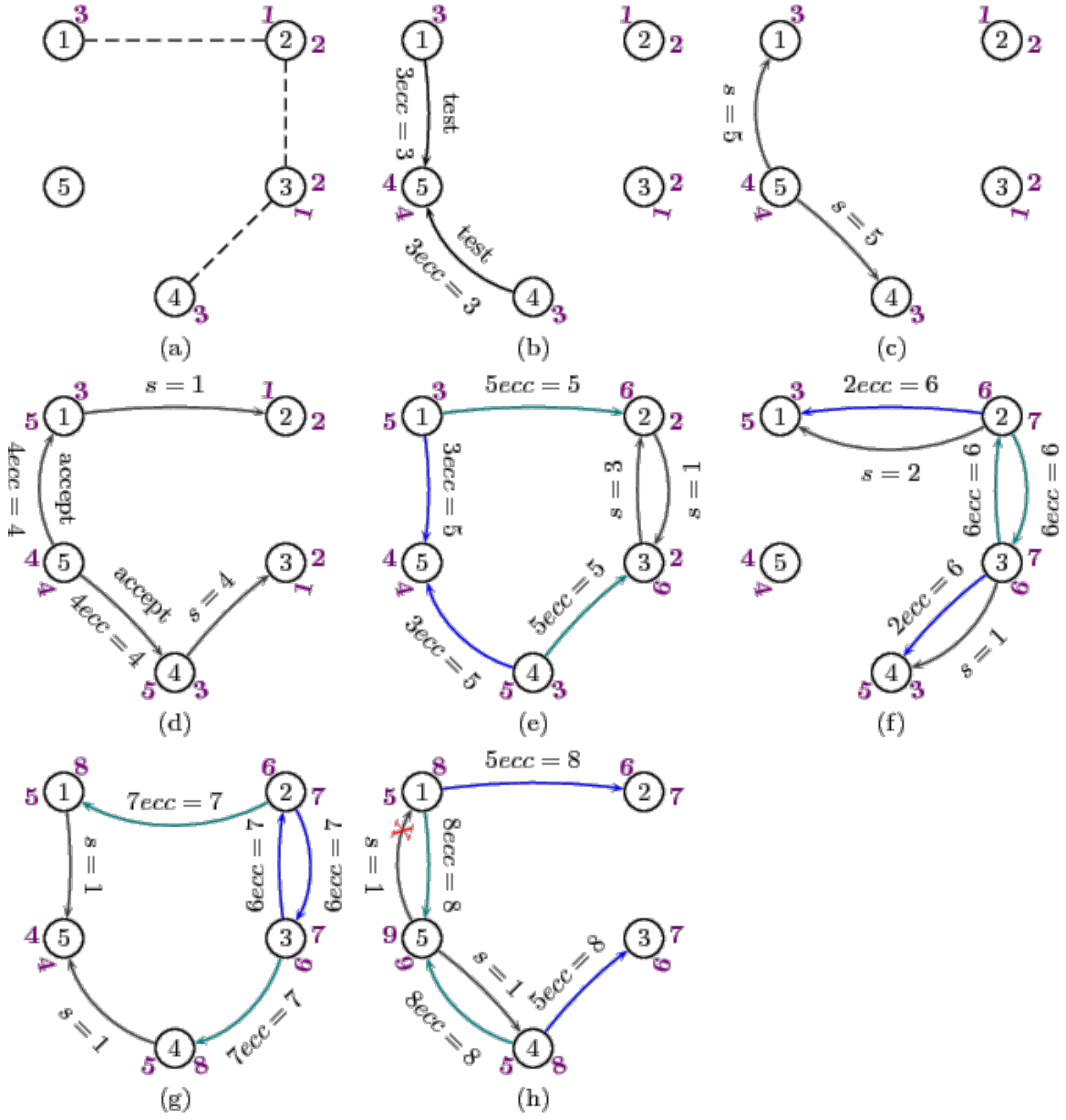

Figure 8.13: Message exchange in a net with five nodes. Before and during the loop breaking.

For six nodes:

When a sixth node arrives to the previous net, the exchange of messages will be as Figure 8.15 and 8.16 show:

In summary for the six nodes, the messages before and during the breaking of the loop are:

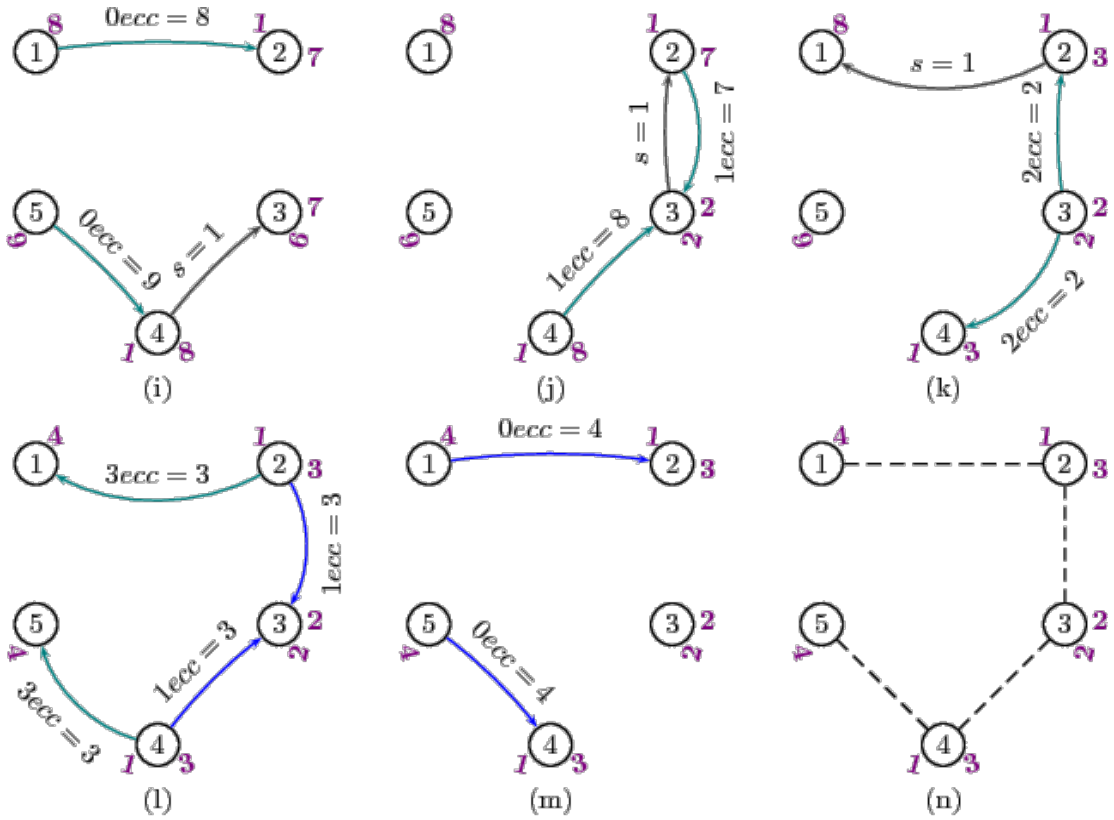

Figure 8.14: Message exchange in a net with five nodes. After the loop breaking.

- 4 *test messages* are generated (in the worst case).
- 4 *accept messages* are generated (in the worst case).
- 14 *init messages*.
- 20 *update messages* are generated, 10 announce a change in the *max2send*, and 10 more are considered under the conditions of the worst case (see the previous chapter).
- 8 *update messages* that inform a change in the *eccentricity*.

After the breaking of the loop the messages are:

- 4 *init messages* are generated.
- 20 *update messages* are generated. 6 announce a change in the *max2send*, and 6 more are considered under the conditions of the worst case.
- 6 *update messages* that inform a change in the *eccentricity*.

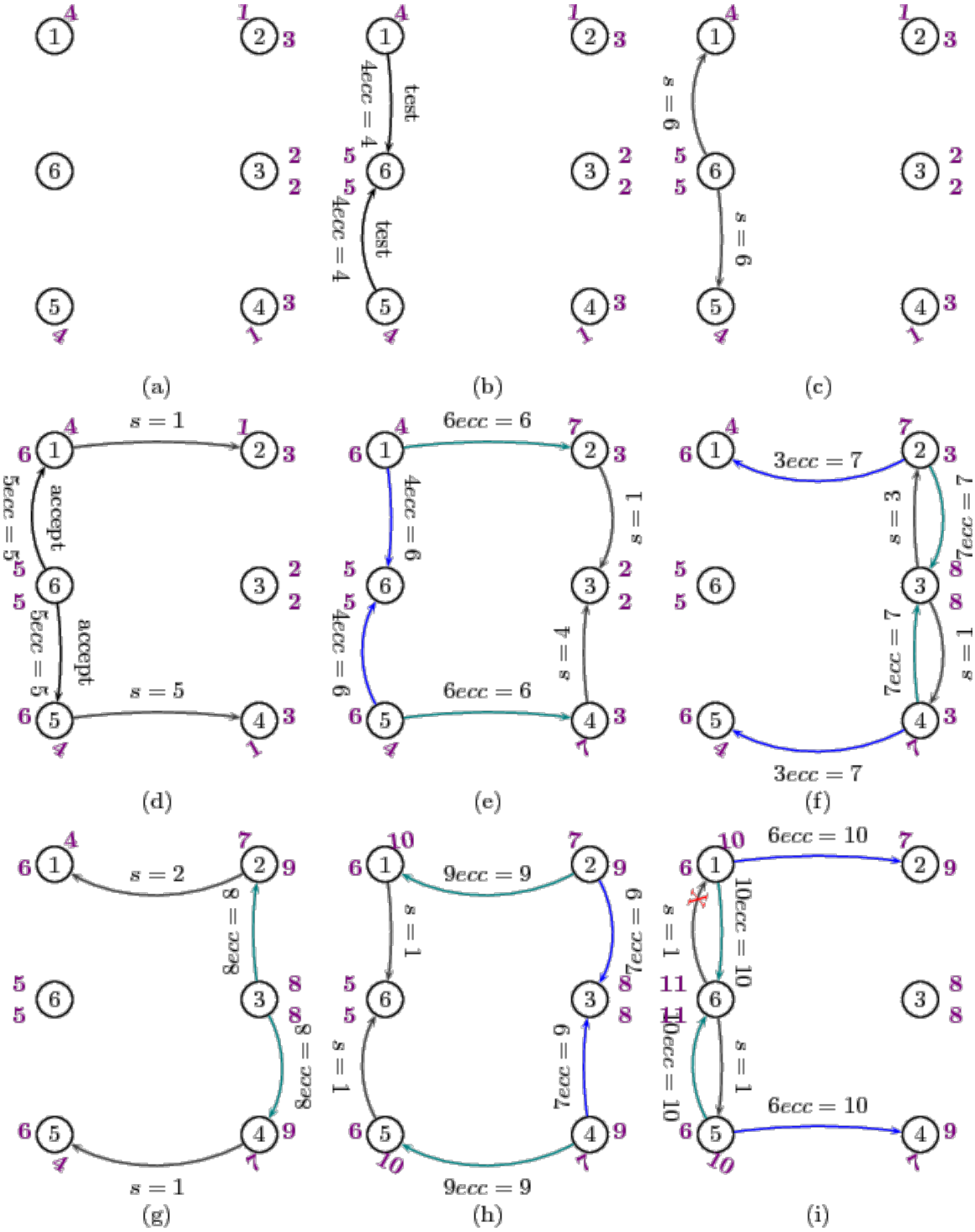

Figure 8.15: Message exchange in a net with six nodes. Before and during the loop breaking.

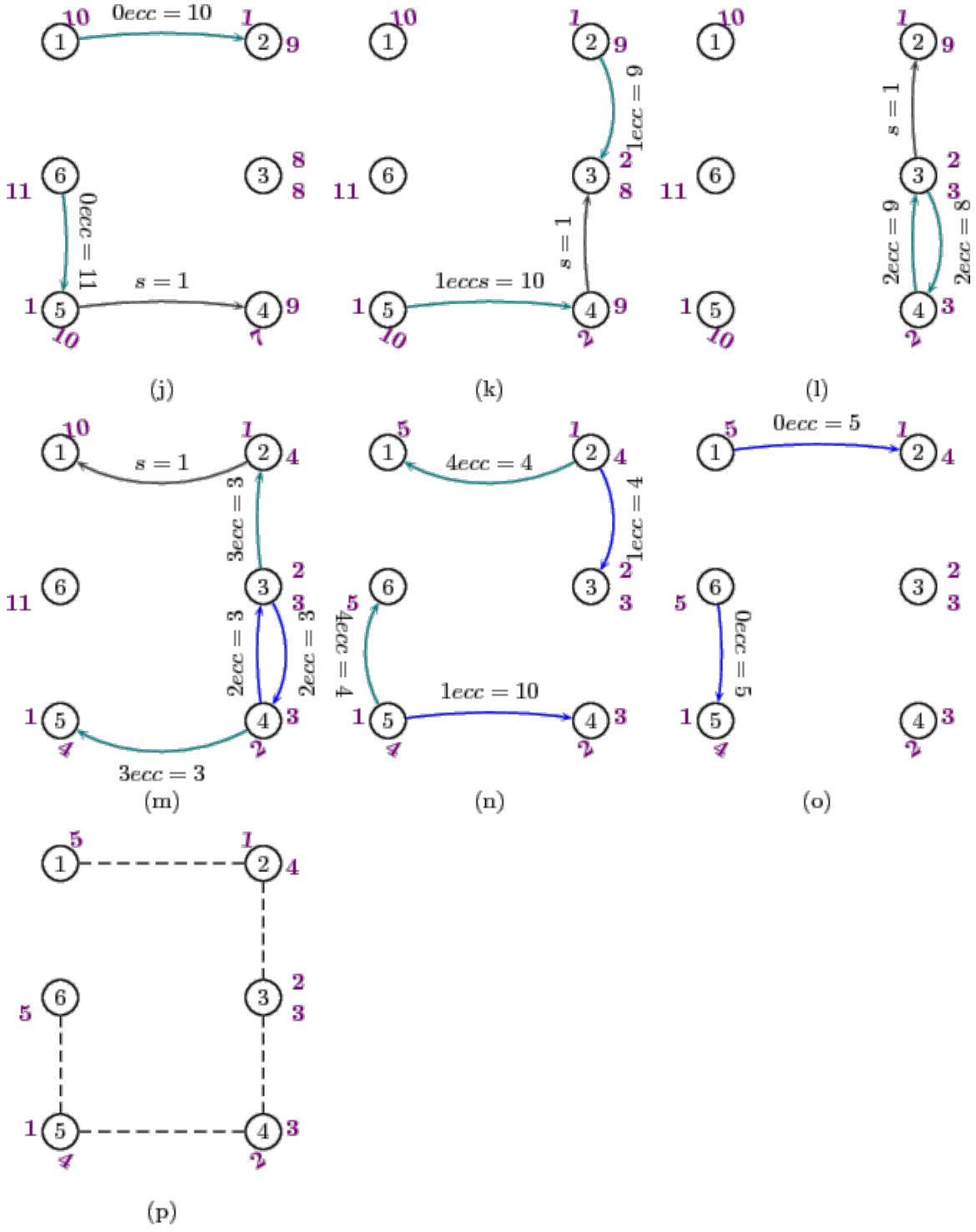

Figure 8.16: Message exchange in a net with six nodes. After the loop breaking.

- 5 *leader messages* are generated.

For seven nodes:

When a seventh node arrives to the previous net, the exchange of messages will be as Figure 8.17, 8.18 and 8.19 show:

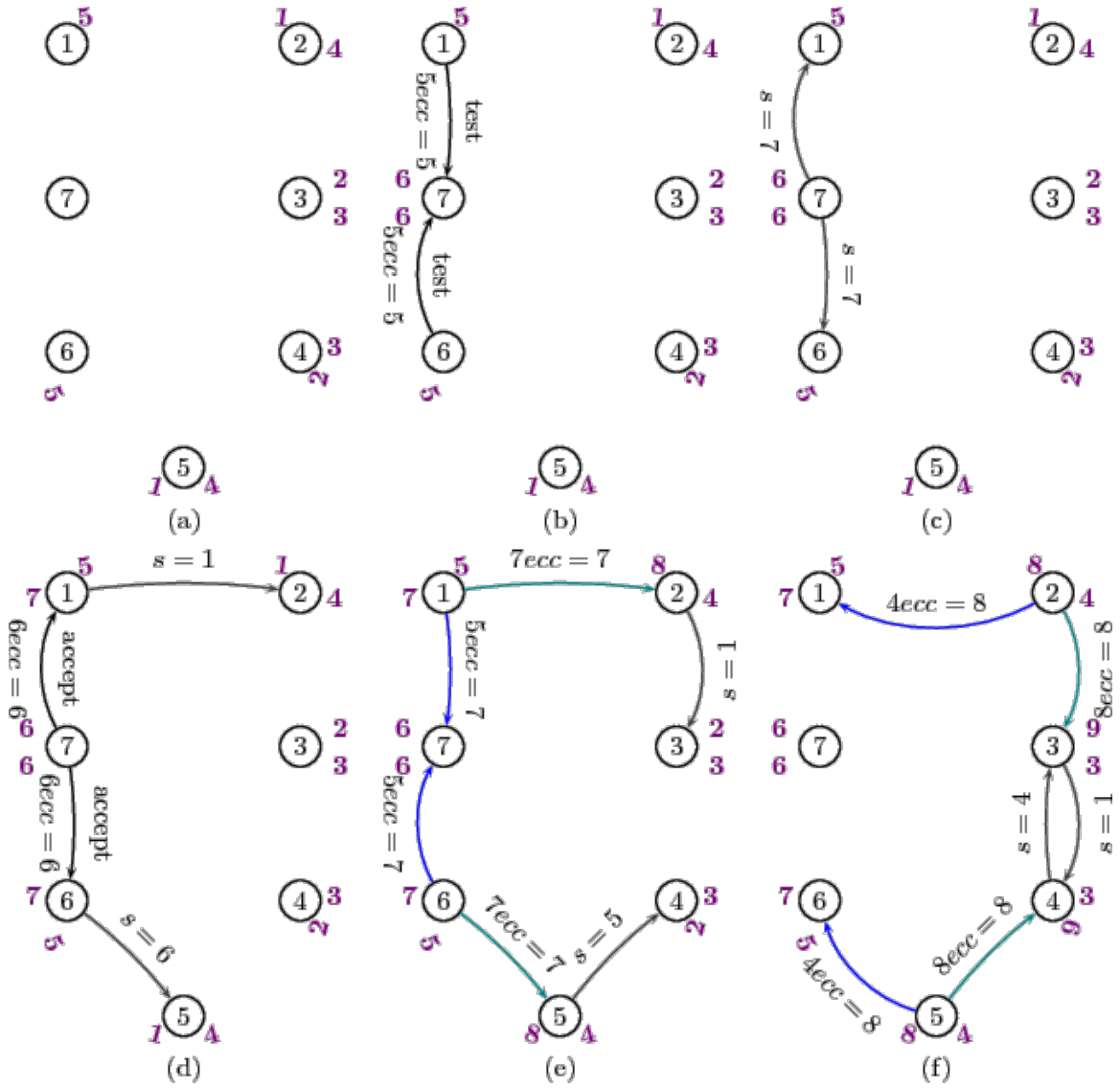

Figure 8.17: Message exchange in a net with seven nodes. Part I.

In summary for the seven nodes, the messages before and during the breaking of the loop are:

- 4 *test messages* are generated (in the worst case).
- 4 *accept messages* are generated (in the worst case).

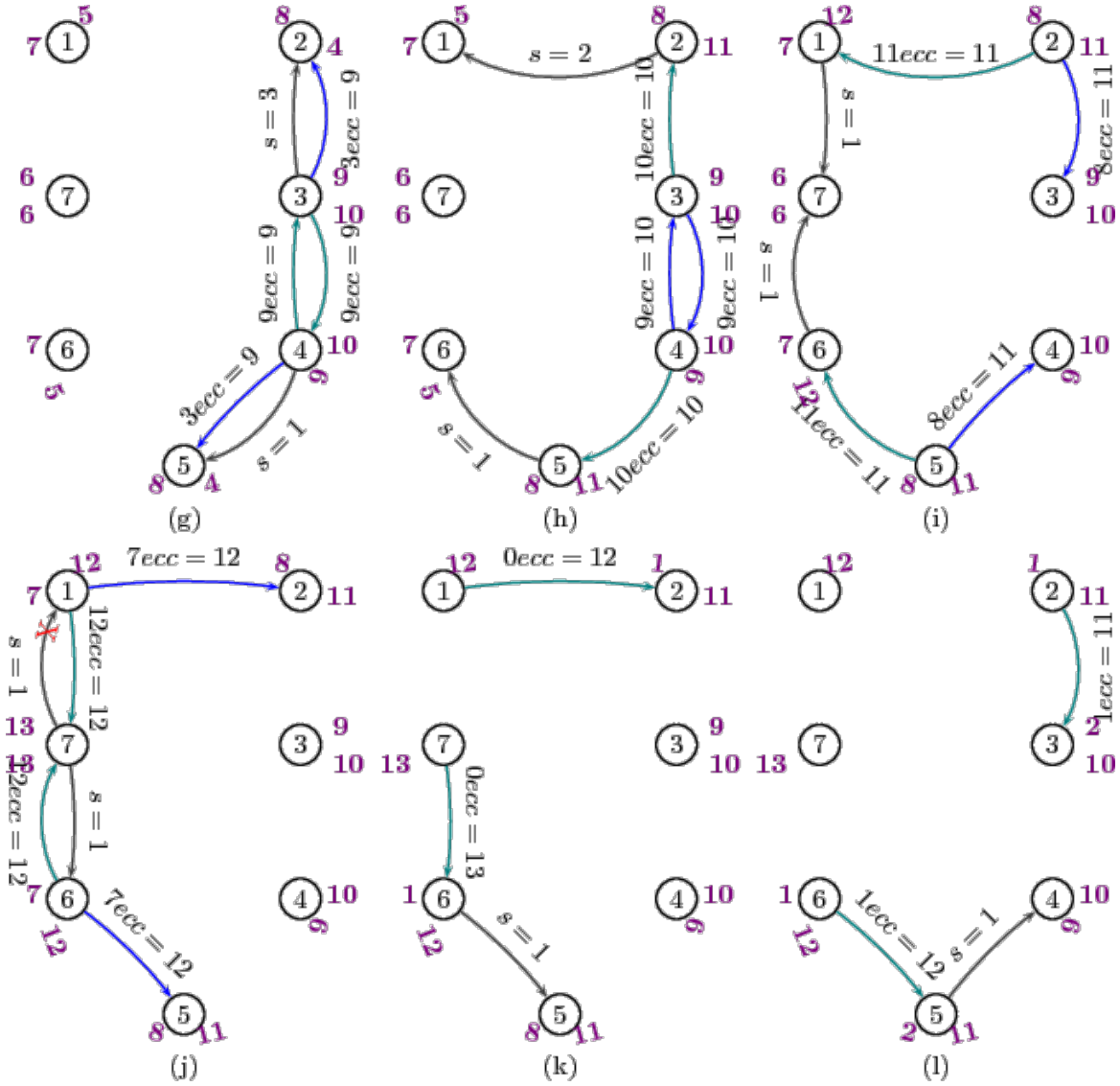

Figure 8.18: Message exchange in a net with seven nodes. Part II.

- 16 *init messages*.
- 24 *update messages* are generated, 12 announce a change in the *max2send*, and 12 more are considered under the conditions of the worst case (see the previous chapter).
- 12 *update messages* that inform a change in the *eccentricity*.

After the breaking of the loop the messages are:

- 5 *init messages* are generated.

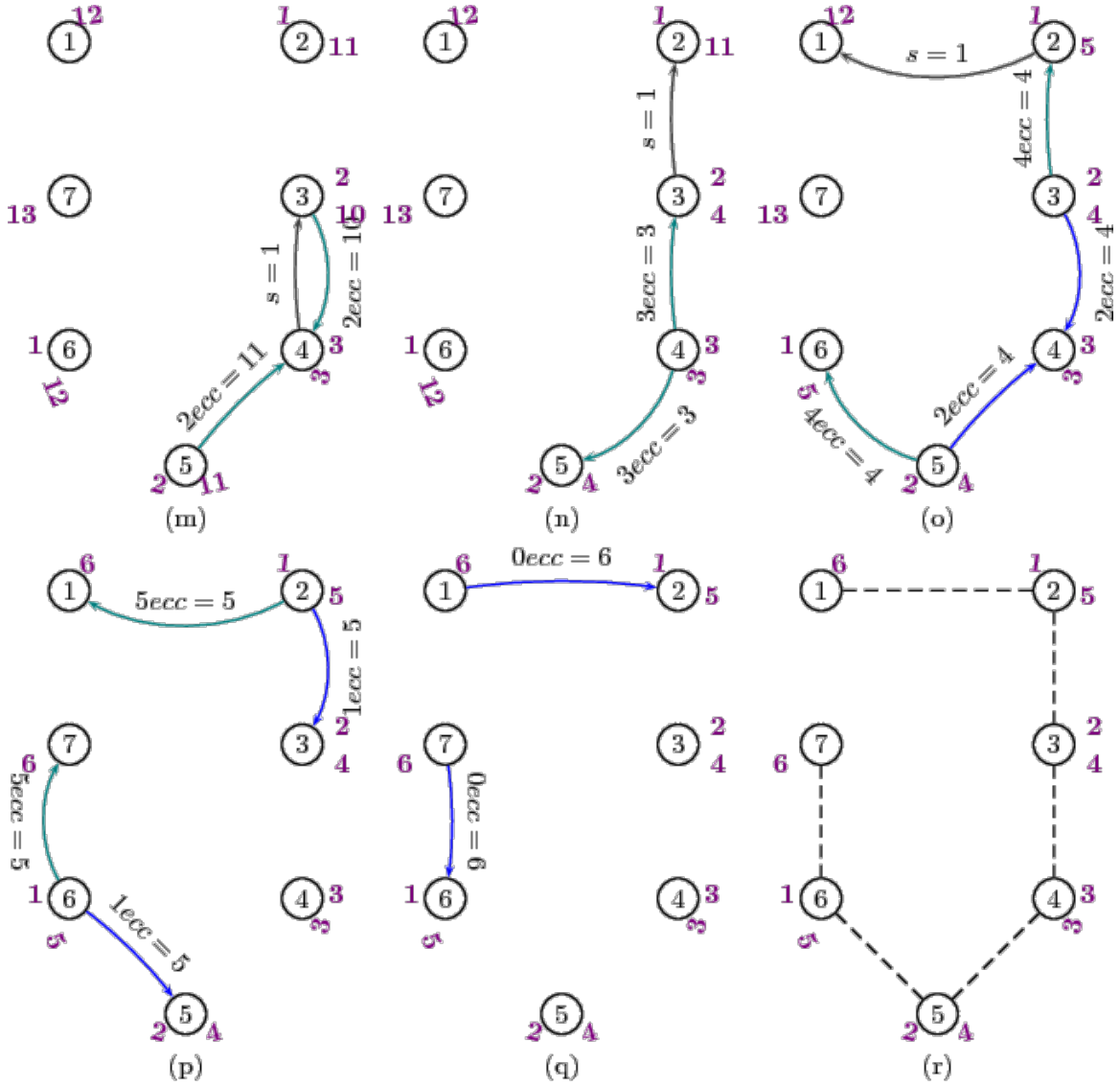

Figure 8.19: Message exchange in a net with seven nodes. Part III.

- 24 *update messages* are generated. 6 announce a change in the *max2send*, and 6 more are considered under the conditions of the worst case.
- 6 *update messages* that inform a change in the *eccentricity*.
- 6 *leader messages* are generated.

For eight nodes:

When a eighth node arrives to the previous net, the exchange of messages will be as Figure 8.20, 8.21, 8.22 and 8.23 show:

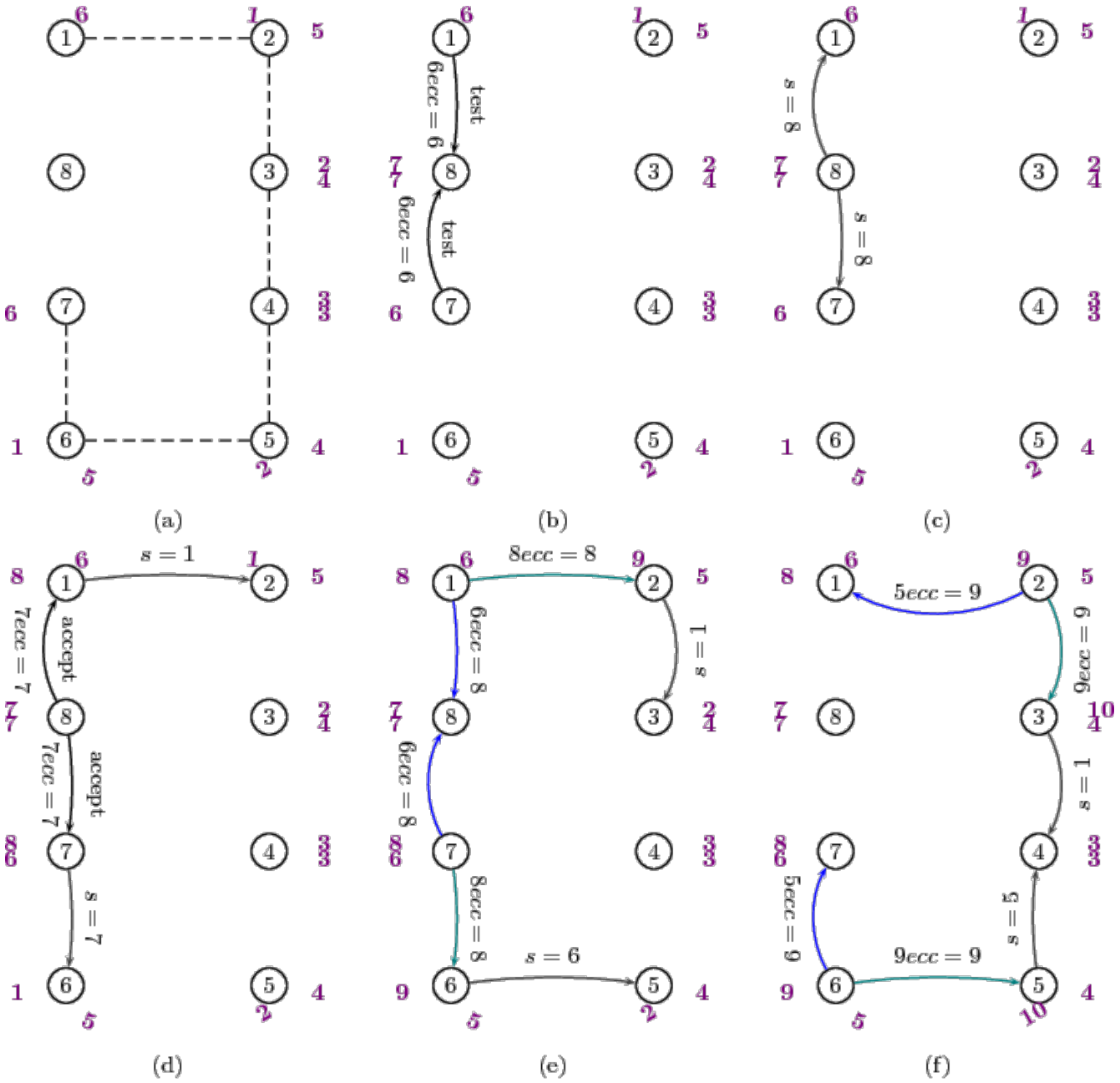

Figure 8.20: Message exchange in a net with eight nodes. Part I.

In summary for the eight nodes, the messages before and during the breaking of the loop are:

- 4 *test messages* are generated (in the worst case).
- 4 *accept messages* are generated (in the worst case).

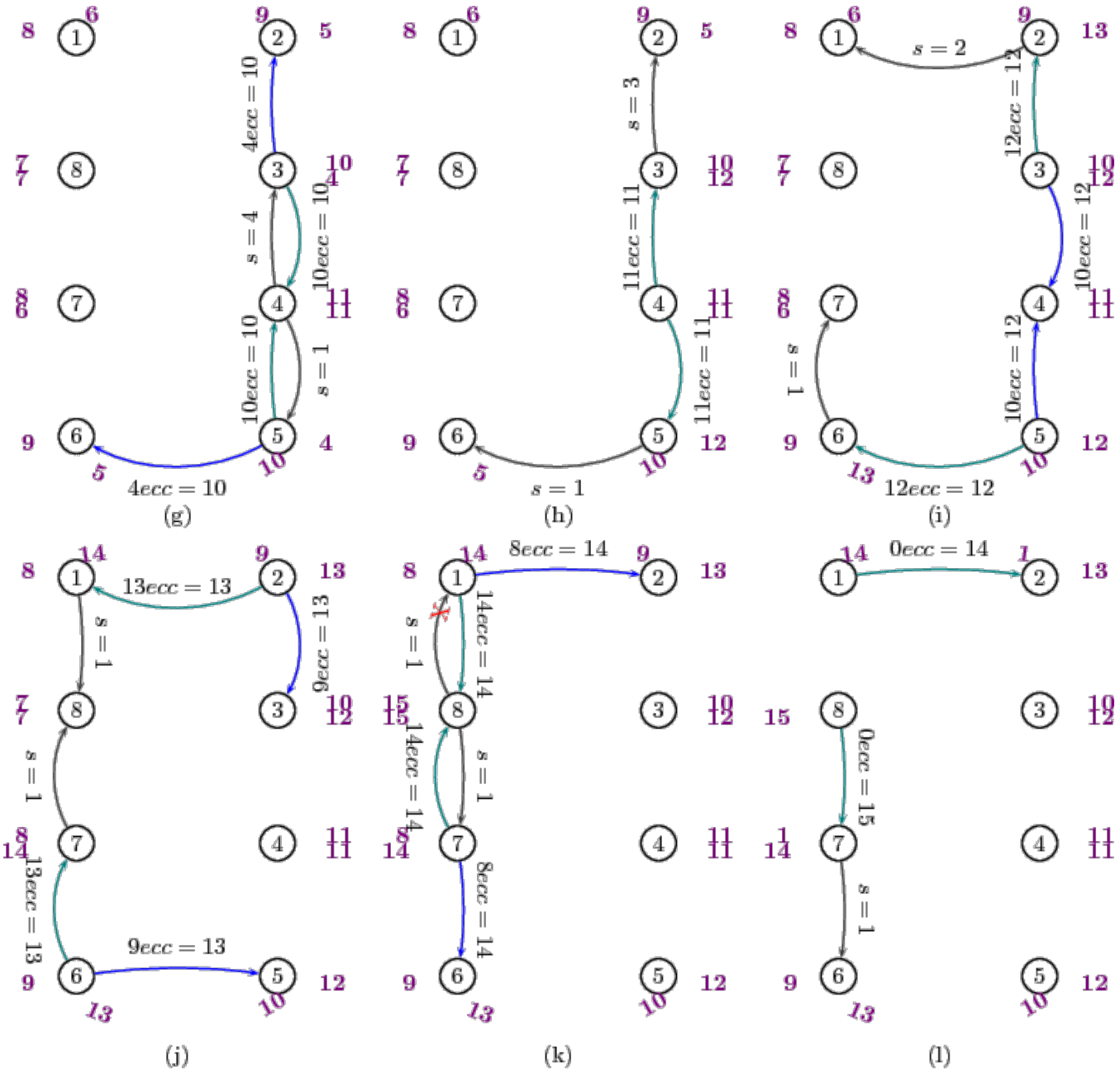

Figure 8.21: Message exchange in a net with eight nodes. Part II.

- 18 *init messages*.
- 28 *update messages* are generated, 14 announce a change in the *max2send*, and 14 more are considered under the conditions of the worst case (see the previous chapter).
- 12 *update messages* that inform a change in the *eccentricity*.

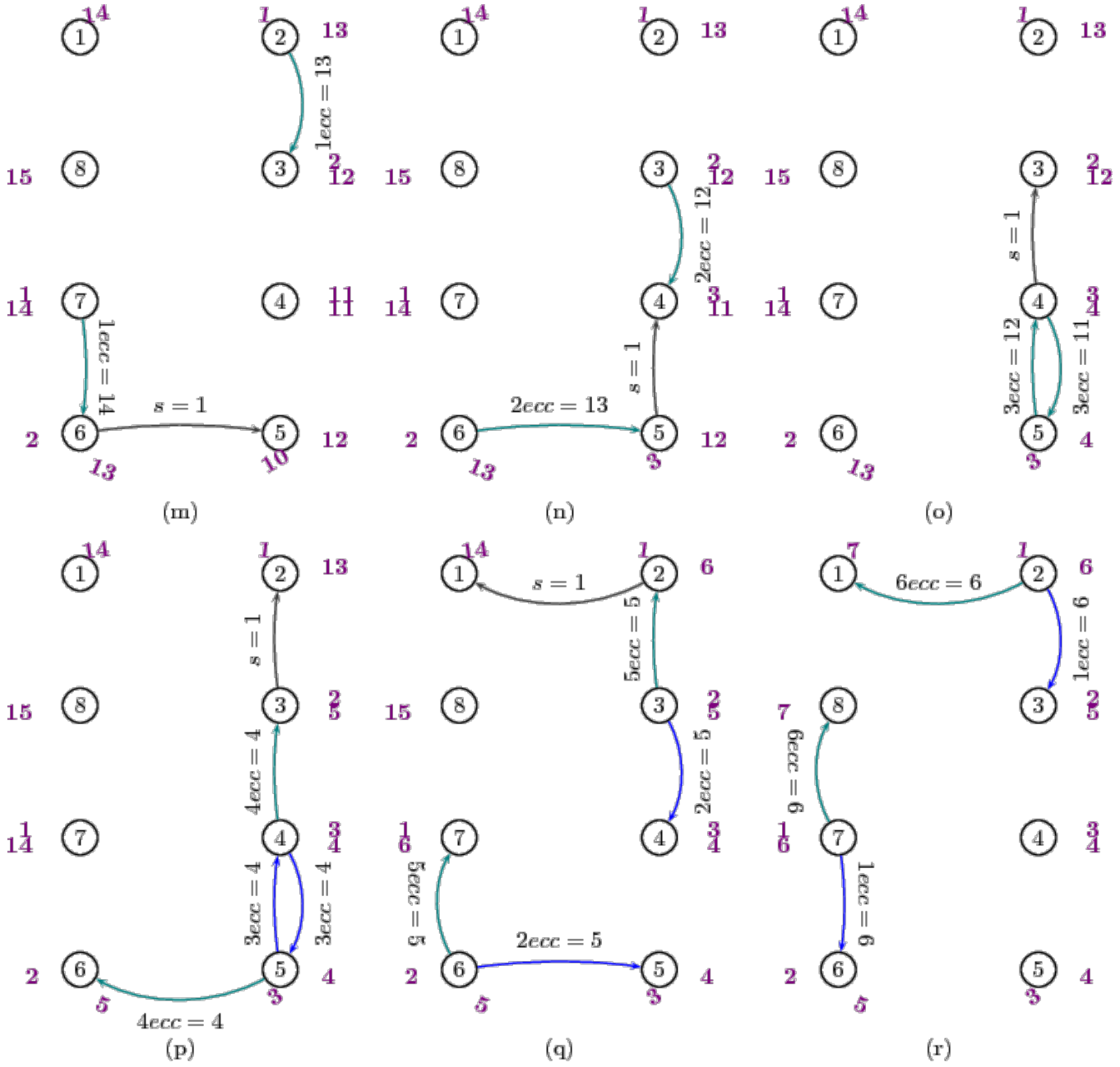

Figure 8.22: Message exchange in a net with eight nodes. Part III.

After the breaking of the loop the messages are:

- 6 *init messages* are generated.
- 28 *update messages* are generated. 6 announce a change in the *max2send*, and 6 more are considered under the conditions of the worst case.

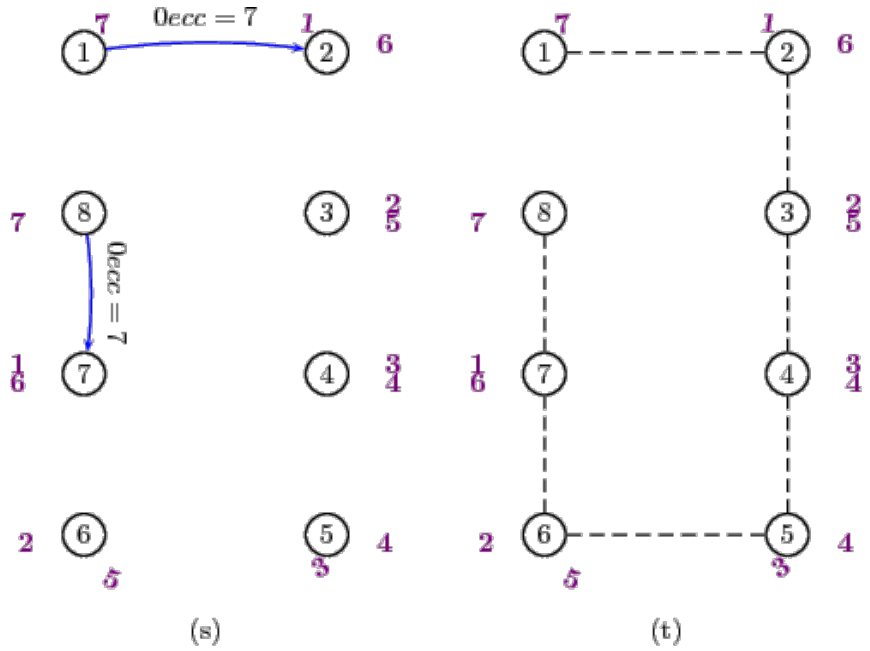

Figure 8.23: Message exchange in a net with eight nodes. Part IV.

- 8 *update messages* that inform a change in the *eccentricity*.
- 7 *leader messages* are generated.

The messages of the eight cases plus one more case are summarized in the Table 8.2 and 8.3.

Table 8.2: Test, Accept, Init and Leader messages for the process of establishment of links and leader calculation.

#_of_nodes	Test	Accept	Init [1]	Leader
1	0	0	0	0
2	2	2	1	1
3	4	4	8	2
4	4	4	10	3
5	4	4	12	4
6	4	4	14	5
7	4	4	16	6
8	4	4	18	7
9	4	4	20	8

[1]Before and during the breaking of the loop
[2]After the breaking of the loop

Table 8.3: Update messages for the process of establishment of links and leader calculation.

#_of_nodes	Init[2]	Update:Ecc[1]	Update:Ecc[2]	Update:Mixed[1]	Update:Mixed[2]
1	0	0	0	0	0
2	0	2	0	0	0
3	1	4	2	8	8
4	2	4	4	12	12
5	3	8	4	16	16
6	4	8	6	20	20
7	5	12	6	24	24
8	6	12	8	28	28
9	7	16	8	32	32

In the best case, the sending of the *leader messages* will be produced only at the end of the establishment of all the possible links. In the worst case, after the establishment of a link there will be enough time to calculate the *leader* of the network and send *leader messages*. So when a new node joins the net, the former *leader* should be deleted and a new one must be calculated. To be consistent in the messages complexity, only the worst cases for each type of messages are taken into account.

First of all, the total number of messages is calculated. The total quantity of messages necessary for the establishment of the links and the leader election in a net of n nodes is represented by the Equation (8.1).

$$TotalMessages(n) = \sum_{i=1}^{n} (T_i + A_i + IB_i + IA_i + UEB_i + UEA_i + UMB_i + UMA_i + L_i) \tag{8.1}$$

Where:

T represents the *test messages*,
A represents the *accept messages*,
IB represents the *init messages* before and during the loop breaking,
IA represents the *init messages* after the loop breaking,
UEB represents the *update messages* that informs the change in the *eccentricity* before and during the loop breaking,
UEA represents the *update messages* that informs the change in the *eccentricity* after the loop breaking,
UMB represents the *update messages* that informs the change in the *max2send* and the change in the *eccentricity* before and during the loop breaking,
UMA represents the *update messages* that informs the change in the *max2send*

and the change in the *eccentricity* after the loop breaking,
L represents the *leader messages*.

For the *test messages*:

$$\sum_{i=1}^{n} T_i = 2 + 4\sum_{i=3}^{n} 4 = 2 + 4\,(n-2) = 4n - 6 \tag{8.2}$$

For the *accept messages*:

$$\sum_{i=1}^{n} A_i = 2 + 4\sum_{i=3}^{n} 4 = 2 + 4\,(n-2) = 4n - 6 \tag{8.3}$$

For the *init messages*:

$$\sum_{i=1}^{n} IB_i = 1 + 2\sum_{i=3}^{n} (i+1) = n^2 + 3n - 9 \tag{8.4}$$

$$\sum_{i=1}^{n} IA_i = \sum_{i=3}^{n} (i-2) = \frac{n^2 - 3n + 2}{2} \tag{8.5}$$

For the *update messages* whose information only updates the *eccentricity*, and are generated before and during the loop breaking:

$$\sum_{i=1}^{n} UEB_i = \begin{cases} 8\sum\limits_{i=1}^{\frac{n-3}{2}} i + 2\,(n-1) = n^2 - 2n + 1 & \text{, if } n \text{ is odd} \\ 8\sum\limits_{i=1}^{\frac{n-2}{2}} i = n^2 - 2n & \text{, if } n \text{ is even} \end{cases} \tag{8.6}$$

For the *update messages* whose information only updates the *eccentricity*, and are generated after the loop breaking:

$$\sum_{i=1}^{n} UEA_i = \begin{cases} 4\sum\limits_{i=1}^{\frac{n-3}{2}} i + 1 + 2 = \dfrac{n^2 - 5}{2} & \text{, if } n \text{ is odd} \\ 4\sum\limits_{i=1}^{\frac{n-4}{2}} i + 1 + 2 + n = \dfrac{n^2 - 4}{2} & \text{, if } n \text{ is even} \end{cases} \tag{8.7}$$

For the *update messages* that informs at the same time the change in the *eccentricity* and in the *max2send*, and are generated before and during the loop breaking:

$$\sum_{i=1}^{n} UMB_i = 4\sum_{i=3}^{n} i - 1 = 2\left(n^2 - n - 2\right) \tag{8.8}$$

For the *update messages* that informs at the same time the change in the *eccentricity* and in the *max2send*, and are generated after the loop breaking:

$$\sum_{i=1}^{n} UMA_i = 4\sum_{i=3}^{n} i - 1 = 2\left(n^2 - n - 2\right) \tag{8.9}$$

For the *leader messages*:

$$\sum_{i=1}^{n} L_i = \sum_{i=1}^{n} (i-1) = \frac{n^2 - n}{2} \tag{8.10}$$

Now replacing (8.2), (8.3), (8.4), (8.5), (8.6),(8.7), (8.8), (8.9) and (8.10) in (8.1) the result is:

$$TotalMessages(n) = \begin{cases} \dfrac{15n^2 + 6n - 63}{2} & \text{, if } n \text{ is odd} \\ \dfrac{15n^2 + 6n - 60}{2} & \text{, if } n \text{ is even} \end{cases} \tag{8.11}$$

The result that is shown in the Equation (8.11) indicates that the proposed algorithm has a message complexity of $O\left(n^2\right)$ under the presence of loops.

At this point is important to have clarity about the structure that is formed at the end of the execution of the algorithm (or partial execution, due to the dynamic topology of the network). This implies to make a short review of some graph theory concepts. See Chapter 5. Remembering that the algorithm permits that every node of the network knows its own eccentricity, the nodes establish communication channels hierarchically (with the neighbour with lower eccentricity and the neighbours with higher eccentricity), loops are avoided, nodes can improve its positions being connected with the neighbour that offers a better eccentricity, and that the *leader* of the network is the node whose eccentricity equals the radius of the graph that represents the network; it can be affirmed that the structure that has been constructed corresponds to a spanning tree of shortest paths (SPT), that is rooted at the central point of the graph, or to be precise a Minimum Diameter Spanning Tree (MDST).

Chapter 9

Algorithm for the Identification of the Nodes to Cover the Entire Network

In Chapter 7 and 8 the strategy to elect the node (that was called *leader*) was explained. The *leader* node is responsible for the correlation of the local events to infer about the state of the global security of the network, and is chosen as the node that is located in a point that minimizes the number of hops from the "local sensors" to the leader, while the stability of the IDS was improved.

For the election of the sensors that need to be spread in the net, the strategy is different. In first place the objective of the existence of these nodes, is the coverage of the entire network, i.e., a set of nodes must be elected in a way that ensure that the packets that cross the network are evaluated by the IDS process; and second, due to the fact that in MANETs the resources are limited, the number of sensors must be minimized.

9.1 Proposed Strategy

The distributive nature of MANETs causes that processes like the routing and broadcasting do not have a predefined path to send messages. In order to solve that situation the creation of a virtual backbone has been proposed by some authors. These backbones have been accomplished computing Dominating Sets (DS) basically in two flavours: Connected Dominating Sets (CDS) and Weakly Connected Dominating Sets (WCDS) [7, 19, 30, 104]. This gives some clues to elect the nodes that will be responsible for the IDS processes guarantying the coverage.

In order to identify a strategy, some definitions must be evaluated:

Dominating Set: a set S is said to be a Dominating Set (of vertices) of a graph $G = (V, E)$ if each node in G is either in S or adjacent to at least one of the nodes in S [20, 7].

Connected Dominating Set: S is a Connected Dominating Set of a graph $G = (V, E)$ if it is a Dominating Set and the resulting subgraph is connected [7], i.e., any pair of nodes in the set can communicate through a path of nodes that are entirely within the CDS.

Weakly Connected Dominating Set: S is a Weakly Connected Dominating Set of a graph $G = (V, E)$ if it is a Dominating Set and all the edges of G have at least one end point in the set [7].

The efficiency of an IDS in a MANET not only can be evaluated by the successful detection of intrusions, but because the resources consumption is controlled. The location of IDS sensors in every node of the network is not very efficient, so a good strategy is to place sensors in certain points without affecting the detection tasks. Given the MDST that is obtained executing the algorithm, a CDS can be constructed; this set will include all the nodes except the leaves of the tree, thus most of the nodes will be executing IDS tasks, but as explained above this solution is not efficient.

In [7] Alzoubi *et ál.,* propose an algorithm to build a Weakly Connected Dominating Set (WCDS) using a colouring strategy over a spanning tree. In the tree, each node is given a rank according to its level. The ranking process is started by the root (whose rank is 0) through the sending of message to its children. When the ranking is completed the root starts the marking by colouring itself as black and sending this information to its children. Each child that receives information that its parent has been marked as black, marks itself as gray, but if the message says that its parent is gray then it will be marked as back. At the end of the algorithm the tree will be compose of gray and black nodes, where the black nodes belong to the WCDS. If this solution is used as strategy to locate the sensors of the IDS, a coverage as the shown in the Figure 9.1 will be obtained.

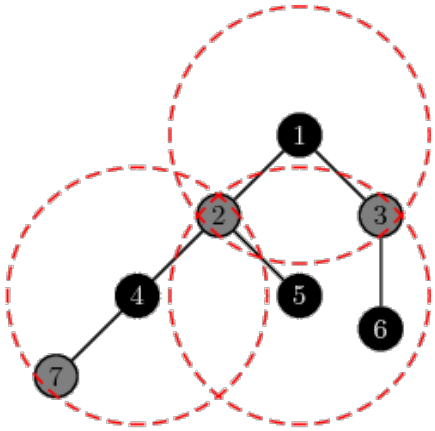

Figure 9.1: Network coverage locating IDS sensors in the WCDS.

Figure 9.1 shows that the IDS sensors are located in *nodes 1*, *4*, *5*, and *6*. *Node 2* packets are analyzed by the sensors of nodes 4, 1, and 5 (see the red circles) which are connected

with the node (black lines indicate the connections). This redundancy in the coverage of a node may be a good strategy in networks with enough resources, but in MANETs it is not efficient. A good coverage for the nodes in MANETs will be as the shown in Figure 9.2.

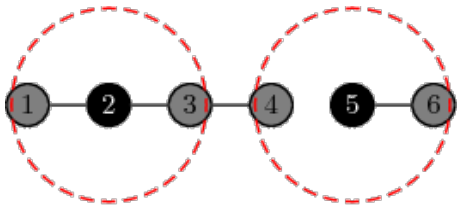

Figure 9.2: Desired coverage of the IDS sensors.

The structure of the Figure 9.2 that is also shown in [7], corresponds to a particular set called *Maximal Independent Set*, the following definitions clarify the term:

Independent Set: a set S is a subset of vertices that is said to be an Independent Set of a graph G if there is no edge between any pair of nodes in S [7, 99]. Here is important to remember Balakrishnan in [13], when affirms that "an independent set is also a dominating set if and only if it is a maximal independent set".

Maximal Independent Set (MIS): A set S is a subset of vertices that is said to be a Maximal Independent Set of a graph G such that if a node is not in the MIS, then it is adjacent to any node of the MIS. This definition implies that the addition of a vertex in the set will break the independence property [19].

The definition of the MIS implies that the shortest distance between any two subsets of vertices in S is either two or three hops [25]. In order to fulfill the objective of efficient use of resources in the MANETs, the strategy to elect the nodes that will monitor the security of themselves and their neighbours, will be to construct a Maximal Independent Set with the largest distance possible. The network drawn in the Figure 9.1 will look as the shown in Figure 9.3 using the MIS approach. Where *node 2* is covered by the monitors located in *node 1* and *5*.

9.2 Design of the Algorithm

The algorithm for the election of the monitors of the IDS will be integrated with the algorithm that was presented in Chapters 7 and 8. Its implementation will require the addition of some conditions to mark the nodes with a "colour", increase the tables with an extra field and add a field to the *leader* and *update messages*. The nodes will be marked with 0, 1, 2, and 3 corresponding to white, black, gray, and purple respectively. The following are the conditions which need to be added:

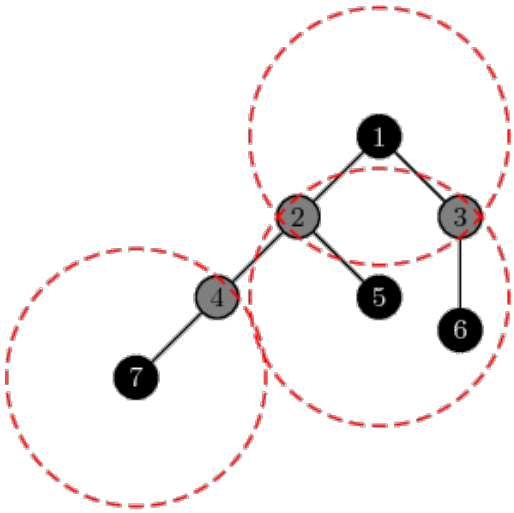

Figure 9.3: Coverage obtained locating the IDS sensors in the MIS of the graph that represents the network.

- If the node has no *leader* yet ($leader\_ID = 0$) then $colour = 0$ (white).
- If the node discovers itself as the leader of the network then $colour = 1$ (black).
- If a message arrives (obeying the conditions to take the information as valid) informing that it is required to delete the current leader, then $colour = 0$ (white).
- If the parent (the neighbour with lower eccentricity, or equal eccentricity and higher ID) is white (related in the table) then the node is white ($colour = 0$).
- If the parent is black then the node is gray ($colour = 2$).
- If the parent is gray, and the node is not in the border of the network then the node is purple ($colour = 3$).
- If the parent is gray, and the node is in the border of the network then the node is black ($colour = 1$).
- If the parent is purple then the node is black ($colour = 1$).

Nodes marked as black are those which monitor the packets that cross its range (including its packets). The colouring and the leader are closely related, so the response to the dynamics of the network will be equal.

The result to apply the algorithm to the network shown in Figure 9.1 is the presented in Figure 9.4.

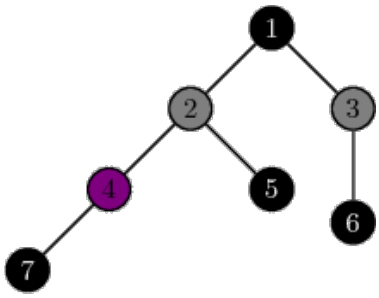

Figure 9.4: Coverage obtained locating the IDS sensors according to the algorithm to elect the monitors.

9.3 Complexity Analysis

Due to the fact that the strategy does not generate more packets, the message complexity does not increase.

Chapter 10

Security for the Algorithm

The algorithm lacks of a mechanism to secure the data and avoid the active or passive attacks against its messages, i.e., it needs some proactive security mechanisms like cryptographic or authentication. Cryptography requires the existence of cryptographic keys and their management. The lack of resources in MANETs makes hard such an implementation [103], but some advances have been made in this field, and the same mechanism used to protect the network can be used also to protect the construction and maintenance of the tree for the IDS.

An interesting strategy, was found in the mechanism proposed by Nargunam and Sebastian [67]. Every node of the network is supposed to have a valid identity. When a node arrives to the network, an identity verification process is initialized by the neighbours and it can be admitted in the net. After the admission into the cluster (the network is divided into clusters) the public keys of the cluster members are exchanged. The packets are encrypted and decrypted in the network layer, but it requires a lot of time because when a node receives the packet it must be decrypted using the private key to know if the message was addressed to the node, if the node is not the final destination, then the message is encrypted using the public key of the next node in the route to the final destination. All the members of the cluster are continuously monitored by the neighbouring nodes, to evaluate the behaviour and deny or allow the entrance to the network. The benefit of the scheme is that the network is divided into clusters, but the clusters do not have clusterhead to the key manage, so a point of failure for the security mechanism is eliminated.

Conclusions, Main Contributions and Future Work

The establishment of a DS in a network and its use in IDS is not new, nor is the fact of constructing minimum diameter spanning trees. The main contribution of this work is the construction and maintenance of a minimum diameter spanning tree in a distributed network where the topology changes frequently.

The algorithm presented is not asynchronous, but for the leader messages a simple strategy of maintain a register of the last leader that was erased helps with the avoidance of invalid information, the same strategy can be used for the other type of messages.

Merge and partitions are events that causes the erasing of the leader of the partitions that are involved, thus the algorithm ensures that the net will never have more than one leader at the same time.

Changes in the topology may imply that while the network is propagating the election of a leader, the election process has to be repeated to elect a leader according to the new conditions. This is not an abnormal behaviour due to the fact that the leader election and the tree construction strictly depend on the network topology.

The algorithm manages packet losses that can be detected through the evaluation of the data contained in the tables of each node, when this happens the nodes are given the capability of request the sending of the message by forwarding the last sent message.

The work presented in [91] constructs a backbone composed of a MIS of 3-hops of distance among the nodes of the set, obtaining a good complexity but the algorithm does not support the dynamic inherent to MANETs and does not break the loops.

The algorithm presented accomplish the construction and maintenance of a minimum diameter spanning tree without calculate and storage the information related with the distance to every node in the net, so the algorithm complete its function with a good performance in the use of the memory resources. The use of the memory and the supporting of the changes in the topology differenciate the work done here with the one presented in [21].

Some algorithms are presented to have a good message complexity, but the calculation assumes that only one node starts the process and the other nodes wake up when receiving messages, and also that the network is static and has already been established. In contrast,

the calculation presented here assumes that in the worst case the nodes are joined to the network one at the time, the nodes that are involved in the merge start the process at the same time, the network can change during the process, and additionally the nodes do not need to wait for the reception of the same number of messages avoiding the existence of dead locks.

In order to improve the stability of the network, the leader can have a backup node, so that a register of the analysis that has been done before can be maintained.

Bibliography

[1] *Internetworking Technology Handbook*, chapter 5: Routing Basics. CISCO.

[2] *Local Area Networks*. The McGraw-Hill Companies, December 2001.

[3] *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, volume 4982/2009, chapter Cross-Layer Optimized Congestion, Contention and Power Control in Wireless Ad Hoc Networks, pages 14–25. Springer Berlin / Heidelberg, May 2008.

[4] *First International Conference on Ad Hoc Networks - ADHOCNETS*, 2009.

[5] S. Ahn and Y. Lim. Manet address configuration using address pool. Internet-draft, Internet Engineering Task Force (IETF), AUTOCONF Working Group, December 2009.

[6] S. Aly and K. Mustafa. Protocol verification and analysis using colored petri nets. Technical Report 04-003, DePaul University and Cairo University, August 2004.

[7] K. Alzoubi, P.-J. Wan, and O. Frieder. Weakly-connected dominating sets and sparse spanners in wireless ad hoc networks. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 96–104, Washington, DC, USA, 2003. IEEE Computer Society.

[8] T. Anantvalee and J. Wu. *Wireless Network Security*, chapter 7: A Survey on Intrusion Detection in Mobile Ad Hoc Networks, pages 159–180. Springer, 2007.

[9] T. R. Andel and A. Yasinsac. Surveying security analysis techniques in manet routing protocols. *IEEE Communications Surveys & Tutorials*, 9(4):70–84, 2007.

[10] P. G. Argyroudis and D. O'Mahony. Secure routing for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7:2–21, 2005.

[11] S. Babu. Security issues in manets. Indo-UK Workshop on Ubiquitous Computing Conference, 2005.

[12] R. Bace and P. Mell. *Intrusion Detection Systems*. National Institute of Standards and Technology - NIST, November 2001. NIST special publication on intrusion detection system.

[13] V. Balakrishnan. *Graph theory*. New York: Schaum's Outline Series, McGraw-Hill, 1997.

[14] P. Baldan, A. Corradini, and U. Montanari. Contextual petri nets, asymmetric event structures, and processes. *Information and Computation*, 171(1):1–49, November 2001.

[15] J. Baras and S. Radosavac. Attacks and defenses utilizing cross-layer interactions in manet. Conference, Workshop on Cross-Layer Issues in the Design of Tactical Mobile Ad Hoc Wireless Networks: Integration of Communication and Networking Functions to Support Optimal Information Management, June 2004.

[16] M. Barile. Graph distance. MathWorld–A Wolfram Web Resource created by Eric W. Weisstein.

[17] F. Bause and P. S. Kritzinger. *Stochastic Petri Net - An Introduction to the Theory*. Vieweg Verlag, 2002.

[18] C. Bernardos, M. Calderon, and H. Moustafa. Survey of ip address autoconfiguration mechanisms for manets. Internet-draft, Internet Engineering Task Force (IETF), AUTOCONF Working Group, November 2008.

[19] J. Blum, 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. The George Washington University.

[20] N. Bray. Dominating set. Web Resource, MathWorld–A Wolfram Web Resource created by Eric W. Weisstein.

[21] M. Bui, F. Butelle, and C. Lavault. A distributed algorithm for constructing a minimum diameter spanning tree. *Journal of Parallel and Distributed Computing*, 64(5):571–577, 2004.

[22] I. Chakeres. Iana allocations for mobile ad hoc network (manet) protocols, rfc 5498. Request for comments, standards track, The Internet Society, Network Working Group, March 2009.

[23] I. Chakeres, J. Macker, and T. Clausen. Mobile ad hoc network architecture. Technical report, Internet Engineering Task Force (IETF), 2007.

[24] Z. Chang, G. Gaydadjiev, and S. Vassiliadis. Routing protocols for mobile ad-hoc networks: Current development and evaluation. In *Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing, ProRisc 2005*, pages 489–494. Dutch Technology Foundation, November 2005.

[25] D. Chen, X. Mao, X. Fei, K. Xing, F. Liu, and M. Song. A convex-hull based algorithm to connect the maximal independent set in unit-disk graphs. In *Wireless Algorithms, Systems, and Applications - Lecture Notes in Computer Science*. Springer Link.

[26] G. Ciardo and R. Zijal. Well-defined stochastic petri nets. In *International Symposium on Modeling, Analysis, and Simulation of Computer Systems*, pages 247–253. IEEE, IEEE Computer Society, 1996.

[27] I. Cidon and O. Mokryn. Propagation and leader election in a multihop broadcast environment. In *12th International Symposium on Distributed Computing (DISC98)*, pages 104–118. Springer-Verlag, 1998.

[28] L. M. S. Committee. Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Standard 802.11, June 1999.

[29] J. Desel and J. Esparza. *Free Choice Petri Nets*. Number 40 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1995.

[30] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 717–724, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[31] T. Fahad, D. Djenouri, and R. Askwith. On detecting packets droppers in manet: A novel low cost approach. In *Third International Symposium on Information Assurance and Security (IAS)*, pages 56–64, August 2007.

[32] A. García and J. Gómez. *Sensor Networks and Configuration: Fundamentals, Standards, Platforms, and Applications*, chapter 17: MANET versus WSN. Springer Berlin Heidelberg, 2007.

[33] S. Giordano. *Handbook of Wireless Networks and Mobile Computing*, chapter Mobile Ad Hoc Networks, pages 325–347. John Wiley & Sons, Inc, 2002.

[34] L. Goodman, A. Lauschke, and E. Weisstein. Dijkstra's algorithm. MathWorld–A Wolfram Web Resource.

[35] J. Haerri and C. Bonnet. On the classification of routing protocols in mobile ad-hoc networks. Technical report, Institut Eurécom, Department of Mobile Communications, 06904 Sophia-Antipolis, France, August 2004.

[36] K. Hangos, R. Lakner, and M. Gerzson. *Intelligent Control Systems: An Introduction with Examples*, volume 60 of *Applied Optimization*, chapter 8: Petri Nets. Kluwer Academic Publishers, 2004.

[37] R. Hassin and A. Tamir. On the minimum diameter spanning tree problem. *Information Processing Letters*, 53(2):109–111, 1995.

[38] M. Heiner, D. Gilbert, and R. Donalson. *Formal Methods for Computational Systems Biology*, volume 5016/2008 of *Lecture Notes in Computer Science*, chapter 8: Petri Nets for Systems and Synthetic Biology, pages 215–264. Springer Berlin / Hildelber, June 2008.

[39] R. Hinden and B. Haberman. Unique local ipv6 unicast addresses, rfc 4193. Request for comments, standards track, The Internet Society, Network Working Group, October 2005.

[40] H. Jasani and K. Yen. Improving the performance of mobile ad hoc networks using directional antennas. *The Annual Review of Communications*, 59:511–520, 2006.

[41] A. Jayasuriya, S. Perreau, A. Dadej, and S. Gordon. Hidden vs exposed terminal problem in ad hoc networks. In F. Safaei, editor, *Australian Telecommunication Networks & Applications Conference (ATNAC)*, pages 52–29, 2004.

[42] C. Jelger. Manet local ipv6 addresses. Internet-draft, Internet Engineering Task Force (IETF), AUTOCONF Working Group, October 2006.

[43] G.-V. Jourdan and G. V. Bochmann. On testing 1-safe petri nets. In *Proceedings of the 2009 Third IEEE International Symposium on Theoretical Aspects of Software Engineering*, pages 275–281. IEEE Computer Society Washington, DC, USA, 2009.

[44] O. Kachirski and R. Guha. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 2*.

[45] P. Karn. Maca - a new channel access method for packet radio. In *9th ARRL Computer Networking Conference*, pages 134–140, 1990.

[46] A. Karygiannis, E. Antonakakis, and A. Apostolopoulos. Detecting critical nodes for manet intrusion detection systems. In *2nd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, pages 7–15, Washington, DC, USA, 2006. IEEE Computer Society.

[47] P. S. Kiran. Protocol architecture for mobile ad-hoc networks using directional antennas. *International Journal of Recent Trends in Engineering (IJRTE)*, 1(1):532–534, May 2009. Issue on Computer Science.

[48] F. Klemm, Z. Ye, S. Krishnamurthy, and S. K. Tripathi. Improving tcp performance in ad hoc networks using signal strength based link management. *Elsevier Ad Hoc Networks*, 3:175–191, 2005.

[49] J. Kniat, W. Jaskowski, K. Jedrzejek, B. Nyczkowski, and S. Skowronek. Life saving system. Final report, Lifetch, Poznan University of Technology, 2004.

[50] Y.-b. Ko and N. Vaidya. CiteSeerX — GeoTORA: A Protocol for Geocasting in Mobile Ad Hoc Networks, 2000.

[51] P. Kuosmanen. Classification of ad hoc routing protocols. Finnish Defence Forces, Naval Academy, 2002.

[52] J. Li, Z. Li, and P. Mohapatra. Aphd: End-to-end delay assurance in 802.11e based manets. In *3rd Annual International Conference on Mobile and Ubiquitous Systems*.

[53] Y. Li and J. Wei. Guidelines on selecting intrusion detection methods in manet. *Proc ISECON 2004*, 2004.

[54] C. Liu and J. Kaiser. A survey of mobile ad hoc network routing protocols. Technical report, University of Magdeburg, October 2005.

[55] H. Liu, X. Jia, P.-J. Wan, X. Liu, and F. Yao. A distributed and efficient flooding scheme using 1-hop information in mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 18(5):658–671, 2007.

[56] J. Liu and S. Singh. Atcp: Tcp for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1300–1315, July 2001.

[57] J. J. N. Liu and I. Chlamtac. *Mobile Ad Hoc Networking*, chapter Mobile Ad Hoc Networking with a View of 4G Wireless: Imperatives and Challenges. John Wiley & Sons, Inc, 2004.

[58] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[59] J. Macker and I. Chakeres. Mobile ad-hoc networks (manet) charter, internet engineering task force. Web page, April 2009.

[60] W. Mallouli, B. Wehbi, and A. Cavalli. Distributed monitoring in ad hoc networks: Conformance and security checking. *Springer-Verlag Berlin Heidelberg*, 2008.

[61] N. Malpani, J. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *DIALM '00: Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 96–103, New York, NY, USA, 2000. ACM.

[62] A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1994.

[63] K. Mase, L. Speakman, and Y. O. and. Manet loop detection for manet. Technical report, Niigata University, November 2007. Internet-Draft.

[64] N. Mohan, R. Duggi, and G. Fan. Patent: Application design framework for manet over bluetooth. Technical report, Samsung Electronics Co., LTD., 2007.

[65] A. Mukherjee, S. Bandyopadhyay, and D. Saha. *Location Management and Routing in Mobile Wireless Networks*, chapter Ad Hoc Wireless Networks. Artech House, Inc, 2003.

[66] T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, April 1989.

[67] A. S. Nargunam and M. P. Sebastian. Distributed security scheme for mobile ad hoc networks. In *Proc IEEE International Conference on Information Acquisition (IEEE ICIA 2006)*, pages 166–171. IEEE, August 2006.

[68] O. Nierstrasz, G. Arévalo, S. Ducasse, R. Wuyts, A. Black, P. Müller, C. Zeidler, T. Genssler, and R. Born. A component model for field devices. In *International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pages 19–26. ACM, ACM, 2002. Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems.

[69] V. Park and J. Macker. Anycast routing for mobile networking. In *Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE*, volume 1, pages 1–5. IEEE, 2002.

[70] J. Peng, L. Cheng, and B. Sikdar. A wireless mac protocol with collision detection. *IEEE Transactions on Mobile Computing*, 6(12):1357–1369, 2007.

[71] S. Radhakrishnan, G. Racherla, and D. Furuno. *Wireless Internet Handbook: Technologies, Standards and Applications*, chapter 16: Mobile Ad Hoc Networks: Principles and Practices, pages 381–402. Auerbach, 2003.

[72] R. Rajaraman. Topology control and routing in ad hoc networks: A survey. *ACM SIGACT News*, 33(2):60–73, June 2002.

[73] W. Reising and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models. Advances in Petri Nets*, volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998.

[74] A. H. M. Rezaul, R. M. A. P. Rajatheva, and K. Ahmed. An efficient collaborative intrusion detection system for manet using bayesian approach. In *MSWiM '06: Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, pages 187–190, New York, NY, USA, 2006. ACM.

[75] F. D. Rosa, V. D. Martino, L. Paglione, and M. Mecella. Mobile adaptive information systems on manet: What we need as basic layer? In *Fourth International Conference on Web Information Systems Engineering Workshop*, pages 260–268, Dec 2003.

[76] P. Santi. *Tutorial: Topology Control in Wireless Ad Hoc Netwoks*. Istituto di Informatica e Telematica del CNR, Pisa, Italy, Tokio, 5th acm mobihoc edition, May 2004.

[77] V. Schlegel. On the problem of the minimum sum of the distances of a point from given points. *Bulleting of the American Mathematical Society*, 1(2):22–52, November 1894.

[78] S. Sen, N. Santhapuri, R. Choudhury, and S. Nelakuditi. Moving away from collision avoidance: Towards collision detection in wireless networks. Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII), ACM, October 2009.

[79] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod. Cross-layer design of ad hoc networks for real-time video streaming. *IEEE Wireless Communications*, 12(4):59–65, August 2005.

[80] C. Shiflet, E. Belding-Royer, and C. Perkins. Address aggregation in mobile ad hoc networks. In *IEEE International Conference on Communications*, volume 6, pages 3734–3738. IEEE, June 2004.

[81] A. Song. Piconet ii, a wireless ad hoc network for mobile handheld devices. Undergraduate Thesis, October 2001.

[82] J. Song, F. Hong, and Y. Guo. A distributed monitoring mechanism for mobile ad hoc networks. In *ISPAN '05: Proceedings of the 8th International Symposium on Parallel Architectures,Algorithms and Network*, pages 236–240, Washington, DC, USA, 2005. IEEE Computer Society.

[83] L. D. M. Soto. Redes de petri: Modelado e implementación de algoritmos para autómatas programables. *Tecnología en Marcha, Tecnológico de Costa Rica*, 21(4):102–125, Octubre 2008.

[84] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C.-Y. Tseng, T. Bowen, K. Levitt, and J. Rowe. A general cooperative intrusion detection architecture for manets. In *Third IEEE International Workshop on Information Assurance*, pages 57–70, 2005.

[85] A. Tanenbaum. *Redes de Computadoras*. Prentice Hall, 2003.

[86] A. Tønnesen. Implementing and extending the optimized link state routing protocol. Master's thesis, University of Oslo, 2004.

[87] University of Bonn, Institute of Computer Science. *5th Workshop on Mobile Ad-Hoc Networks - WMAN 2009*, 2009.

[88] R. van Glabbeek, U. Goltz, and J.-W. Schicke. Symmetric and asymmetric asynchronous interaction. In *Electronic Notes in Theoretical Computer Science*, volume 229, pages 77–95. Elsevier Science Publishers, July 2009. Proceedings of the First Interaction and Concurrency Experiences Workshop (ICE 2008).

[89] I. Vidal, C. García, I. Soto, and J. Moreno. Servicios de valor añadido en redes móviles ad-hoc. *Telecom I+D*, 2004.

[90] H. Wang, B. Crilly, W. Zhao, C. Autry, and S. Swank. Implementing mobile ad hoc networking (manet) over legacy tactical radio links. In *IEEE Military Communications Conference, 2007. MILCOM 2007*, pages 1–7, October 2007.

[91] Y. Wang, W. Wang, and X.-Y. Li. Distributed low-cost backbone formation for wireless ad hoc networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 2–13, New York, NY, USA, 2005. ACM.

[92] E. Weisstein. Central point. MathWorld–A Wolfram Web Resource.

[93] E. Weisstein. Central point. MathWorld–A Wolfram Web Resource.

[94] E. Weisstein. Centroid point. MathWorld–A Wolfram Web Resource.

[95] E. Weisstein. Graph. MathWorld–A Wolfram Web Resource.

[96] E. Weisstein. Graph center. MathWorld–A Wolfram Web Resource.

[97] E. Weisstein. Graph diameter. MathWorld–A Wolfram Web Resource.

[98] E. Weisstein. Graph eccentricity. MathWorld–A Wolfram Web Resource.

[99] E. Weisstein. Independent set. MathWorld–A Wolfram Web Resource.

[100] E. Weisstein. Minimum spanning tree. MathWorld–A Wolfram Web Resource.

[101] E. Weisstein. Spanning tree. MathWorld–A Wolfram Web Resource.

[102] B. Wu, J. Chen, J. Wu, and M. Cardei. *Wireless Network Security*, chapter 5: A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks, pages 103–135. Springer, 2007.

[103] B. Wu, J. Wu, and M. Cardei. *Handbook of Research on Wireless Security*, chapter A Survey of Key Management in Mobile Ad Hoc Networks, pages 479–499. 2008.

[104] J. Wu, W. Lou, and F. Dai. Extended multipoint relays to determine connected dominating sets in manets. *IEEE Transactions on Computers*, 55(3):334–347, 2006.

[105] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications*, 11(1):38–47, 2004.

[106] R. Yang, S. Zhou, and C. Fan. A new algorithm for network diameter. In *The 9th International Conference for Young Computer Scientists*, pages 247–252. IEEE Computer Society, November 2008.

[107] B. Ye and K.-M. Chao. *Spanning Trees and Optimization Problems*. Chapman and Hall, 2004.

[108] J. Yu and P. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 7(1):32–48, 2005.

[109] S. Zaki, M. Ngadi, and A. Razak. A review of dalay aware routing protocols in manet. *ISSR Journals, Ccomputer Sciences Letters*, 1(1):1–12, June 2009.

[110] H. Zhai, J. Wang, X. Chen, and Y. Fang. Medium access control in mobile ad hoc networks: Challenges and solutions. *Wireless Communications & Mobile Computing*, 6(2):151 – 170, 2006.

[111] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283, New York, NY, USA, 2000. ACM.

[112] L. Zhao. *Topology Control for Mobile Ad Hoc Networks*. PhD thesis, University of Delaware, 2007.

[113] H. Zhou and S. Singh. Content based multicast (cbm) in ad hoc networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 51–60. IEEE Press, 2000.